

THE DINI GROUP

LOGIC Emulation Source

User Guide

DN6000K10PCI

LOGIC EMULATION SOURCE

DN6000K10PCI User Manual Version 1.2

© The Dini Group

1010 Pearl Street • Suite 6

La Jolla, CA92037

Phone 858.454.3419 • Fax 858.454.1279

support@dinigroup.com

www.dinigroup.com

Table of Contents

ABOUT THIS MANUAL	1
1 MANUAL CONTENTS.....	1
2 ADDITIONAL RESOURCES	2
3 CONVENTIONS	2
3.1 <i>Typographical</i>	2
3.2 <i>Online Document</i>	4
4 RELEVANT INFORMATION	4
GETTING STARTED	6
1 PRECAUTION	6
2 THE DN6000K10PCI LOGIC EMULATION KIT	6
3 INSTALLATION INSTRUCTIONS	8
3.1 <i>Jumper Setup</i>	8
3.2 <i>Jumper Description</i>	10
3.3 <i>Switch Setup and Description</i>	12
3.4 <i>Oscillator Setup</i>	12
3.5 <i>PPC RS232 Port Setup</i>	13
3.6 <i>Powering ON the DN6000K10PCI</i>	13
4 PLAYING WITH YOUR DN6000K10PCI VIA AETEST	14
5 PLAYING WITH YOUR DN6000K10PCI VIA THE USB INTERFACE.....	17
6 PLAYING WITH YOUR DN6000K10PCI VIA THE PPC'S.....	18
INTRODUCTION TO USB CONTROLLER SOFTWARE	19
1 EXPLORING THE SOFTWARE TOOLS	19
1.1 <i>USBController</i>	19
1.1.1 <i>Getting Started with USBController</i>	20
1.1.2 <i>Basic Menu Operations</i>	20
1.1.3 <i>Enable/Disable USB to FPGA Communication</i>	21
1.1.4 <i>File Menu</i>	22
1.1.5 <i>Edit Menu</i>	22
1.1.6 <i>FPGA Configuration Menu</i>	22
1.1.7 <i>FPGA MemoryMenu</i>	23
1.1.8 <i>Settings/Info Menu</i>	24
1.2 <i>PCI AETEST Application</i>	25
INTRODUCTION TO VIRTEX-II PRO AND ISE	27
1 VIRTEX-II PRO.....	27
1.1 <i>Summary of Virtex-II Pro Features</i>	27
1.2 <i>PowerPC™ 405 Core</i>	28
1.3 <i>RocketIO 3.125 Gbps Transceivers</i>	28
1.4 <i>Virtex-II FPGA Fabric</i>	29
2 FOUNDATION ISE 6.1i	31
2.1 <i>Foundation Features</i>	31
2.1.1 <i>Design Entry</i>	31
2.1.2 <i>Synthesis</i>	32
2.1.3 <i>Implementation and Configuration</i>	32
2.1.4 <i>Board Level Integration</i>	33
3 VIRTEX-II PRO DEVELOPER'S KIT	33
INTRODUCTION TO THE REFERENCE DESIGN	35

1	EXPLORING THE REFERENCE DESIGN	35
1.1	What is the Reference Design?	35
1.2	Using the Reference Design.....	36
1.3	Compiling the Reference Design.....	38
1.3.1	The Xilinx Embedded Development Kit (EDK)	38
1.3.2	Synplicity Synplify.....	38
1.3.3	Xilinx ISE.....	38
1.3.4	The Build Utility: Make.bat	38
2	GETTING MORE INFORMATION	45
2.1	Printed Documentation	45
2.2	Electronic Documentation	45
2.3	Online Documentation	45
	PROGRAMMING/CONFIGURING THE HARDWARE.....	46
1	PROGRAMMING THE CONFIGURATION FPGA	46
2	MCU DETAILS / PROGRAMMING THE MCU	50
3	CONFIGURING HYPERTERMINAL	51
4	CONFIGURING THE FPGA USING SELECTMAP.....	52
4.1	Bit File Generation for SelectMAP Configuration.....	53
4.2	Creating Configuration File “main.txt”	57
4.2.1	Verbose Level	57
4.2.2	Sanity Check	58
4.2.3	Format of “main.txt”	58
4.3	Starting SelectMAP Configuration	60
4.3.1	Description of Main Menu Options	61
4.4	Bitstream Encryption	64
	BOARD HARDWARE	65
1	INTRODUCTION TO THE BOARD	65
1.1	DN6000K10PCI Functionality	65
2	VIRTEX-II PRO FPGA.....	66
2.1	FPGA (2VP70) Facts	66
3	FPGA CONFIGURATION	67
3.1	Micro Controller Unit (MCU)	67
3.1.1	MCU EEPROM Interface	68
3.1.2	MCU SRAM External.....	68
3.1.3	MCU FLASH	69
3.1.4	MCU USB 2.0 Interface.....	69
3.1.5	RS232 Interface.....	70
3.2	Configuration FPGA.....	71
3.2.1	Configuration PROM/FPGA Programming	72
3.2.2	Design Notes on the Configuration FPGA	73
3.3	SmartMedia	74
3.3.1	SmartMedia Connector	74
3.3.2	SmartMedia connection to Spartan (Configuration FPGA)/MCU.....	75
3.4	Boundary-Scan (JTAG, IEEE 1532) Mode.....	76
3.4.1	FPGA JTAG Connector	76
3.4.2	FPGA JTAG connection to Configuration FPGA	76
4	CLOCK GENERATION	77
4.1	Clock Methodology	77
4.2	Clock Source Jumpers.....	81
4.2.1	Clock Source Jumper Header.....	82
4.3	Roboclocks	82
4.3.1	RoboClock PLL Clock Buffers.....	82
4.3.2	RoboClock Configuration Jumpers.....	84
4.3.3	Roboclock Configuration Headers.....	88
4.3.4	Useful Notes and Hints	88
4.3.5	Customizing the Oscillators	89
4.3.6	Common Clock Source Selections.....	89
4.4	External Clocks	89
4.4.1	External SMA Clock.....	90
4.4.2	Connections between FPGA’s and External SMA Clock Buffer	90
4.5	DDR Clocking	90
4.5.1	Clocking Methodology	91
4.5.2	Connections between FPGA’s and DDR PLL Clock Buffer	92

4.6	Power PC (PPC) Clock – Sytem Clock.....	93
4.6.1	Clocking Methodology	93
4.6.2	Connections between FPGA's and System Clock Buffer	93
4.7	Rocket IO Programmable Clocks	94
4.7.1	Clocking Methodology	94
4.7.2	ICS8442 Programmable LVDS Clock Synthesizer.....	95
4.7.3	Connections between FPGA's and RocketIO Clock Synthesizers.....	95
5	RESET TOPOLOGY.....	96
5.1	DN6000K10PCI Reset	96
5.2	PPC Reset.....	97
6	MEMORY	98
6.1	Synchronous SRAM.....	98
6.1.1	SSRAM Configuration.....	102
6.1.2	SSRAM Clocking.....	103
6.1.3	SRAM Termination.....	103
6.1.4	SRAM Connection to the FPGA's.....	103
6.2	DDR SDRAM.....	119
6.2.1	Basics of DDR Operation	119
6.2.2	DDR SDRAM Configuration	119
6.2.3	DDR SDRAM Clocking	120
6.2.4	DDR SDRAM Termination	120
6.2.5	DDR SDRAM Power Supply	122
6.2.6	DDR SDRAM Connection to the FPGA	122
7	ROCKET IO TRANSCEIVERS.....	143
7.1	SMB Connectors.....	144
7.1.1	FPGA to SMB Connector	144
8	CPU DEBUG AND CPU TRACE.....	146
8.1	CPU Debug	147
8.1.1	CPU Debug Connectors	148
8.1.2	CPU Debug Connection to FPGA's	148
8.2	CPU Trace.....	149
8.2.1	CPU Trace Connectors.....	149
8.2.2	Combined CPU Trace/Debug Connection to FPGA's	150
9	GPIO LED's.....	152
9.1	Status Indicators.....	152
9.2	FPGA A GPIO LED's.....	153
10	PCI INTERFACE.....	154
10.1	Connection to the FPGA.....	155
10.1.1	PCI VCCO on the FPGA	155
10.1.2	PCI Edge Connector.....	155
10.1.3	Connection between the PCI connector and the FPGA.....	156
10.2	PCI/PCI-X Hardware Setup.....	161
10.2.1	Present Signals	161
10.2.2	M66EN and PCIxCAP Encoding.....	162
10.2.3	Further Information on PCI/PCI-X Signals.....	163
11	POWER SYSTEM	163
11.1	Stand Alone Operation.....	163
11.1.1	External Power Connector	164
11.1.2	Power Monitors	165
11.1.3	Power Indicators.....	165
12	TEST HEADER & DAUGHTER CARD CONNECTIONS.....	166
12.1	Test Header	166
12.1.1	Test Header Connector.....	168
12.1.2	Test Header Pin Numbering.....	168
12.2	DN3000K10SD Daughter Card.....	169
12.2.1	Daughter Card LED's	171
12.2.2	Power Supply	172
12.2.3	Unbuffered IO	173
12.2.4	Buffered IO	173
12.2.5	LVDS IO	173
12.2.6	Connection between FPGA and the Daughter Card Headers.....	174
13	MECHANICAL.....	198
13.1.1	PWB Dimension.....	198
APPENDIX A – ADDRESS MAPS		200
FPGA A		201

FPGA B	202
FPGA C	204
FPGA D	205
FPGA E	206
FPGA F	207

APPENDIX B - AETEST 209

1	AETEST INSTALLATION INSTRUCTIONS	209
1.1	DOS and Windows 95/98/ME using DPMI	209
1.2	Windows 98/ME using a VxD driver	209
1.3	Windows 2000/XP	210
1.4	Windows NT	210
1.5	Linux	211
1.6	Solaris	212
2	AETEST BASIC C++ FUNCTIONS	213
2.1	bar_write_byte	213
2.1.1	Description	213
2.1.2	Arguments	213
2.1.3	Return Values	213
2.1.4	Notes	213
2.2	bar_write_word	214
2.2.1	Description	214
2.2.2	Arguments	214
2.2.3	Return Values	214
2.2.4	Notes	214
2.3	bar_write_dword	215
2.3.1	Description	215
2.3.2	Arguments	215
2.3.3	Return Values	215
2.3.4	Notes	215
2.4	bar_read_byte	216
2.4.1	Description	216
2.4.2	Arguments	216
2.4.3	Return Values	216
2.4.4	Notes	216
2.5	bar_read_word	217
2.5.1	Description	217
2.5.2	Arguments	217
2.5.3	Return Values	217
2.5.4	Notes	217
2.6	bar_read_dword	218
2.6.1	Description	218
2.6.2	Arguments	218
2.6.3	Return Values	218
2.6.4	Notes	218
2.7	dma_buffer_allocate	219
2.7.1	Description	219
2.7.2	Arguments	219
2.7.3	Return Values	219
2.7.4	Notes	219
2.8	dma_buffer_free	220
2.8.1	Description	220
2.8.2	Arguments	220
2.8.3	Return Values	220
2.8.4	Notes	220
2.9	dma_write_dword	221
2.9.1	Description	221
2.9.2	Arguments	221
2.9.3	Return Values	221
2.9.4	Notes	221
2.10	dma_read_dword	222
2.10.1	Description	222
2.10.2	Arguments	222
2.10.3	Return Values	222
2.10.4	Notes	222
2.11	pci_rdw	223

2.11.1	Description	223
2.11.2	Arguments	223
2.11.3	ReturnValues	223
2.11.4	Notes.....	224
2.12	<i>DeviceIoControl</i>	225
2.12.1	Description	225
2.12.2	Arguments	225
2.12.3	Return Values	225
2.12.4	Notes.....	226
2.12.5	Derived Functions	227

List of Figures

Figure 1 - DN6000K10PCI LOGIC Emulation Board.....	7
Figure 2 - Default Jumper Setup.....	9
Figure 3 DN6000k10PCI Not Found.....	20
Figure 4: Booting from FLASH.....	20
Figure 5: Main USBController Screen	21
Figure 6 - New Project Screen Shot	53
Figure 7 - Input File.....	54
Figure 8: New Project Dialog Box	54
Figure 9: Project Navigator.....	55
Figure 10 - Main Menu	61
Figure 11 - Interactive Configuration Option Menu.....	63
Figure 12 - DN6000K10PCI Block Diagram	65
Figure 13 - MCU EEPROM Interface	68
Figure 14 - MCU SRAM	69
Figure 15 - MCU FLASH	69
Figure 16 - USB Connector	70
Figure 17 - MCU Serial Port.....	70
Figure 18 – Configuration PROM/FPGA Programming Header.....	73
Figure 19 - SmartMedia Connector.....	75
Figure 20 - FPGA JTAG Connector	76
Figure 21 - Clocking Block Diagram.....	77
Figure 22 - LVPECL Clock Input and Termination	82
Figure 23 - Clock Source Jumper.....	82
Figure 24 - RoboClock Functional Block Diagram	84
Figure 25 - RoboClock Configuration Jumpers	88
Figure 26 - External SMA Clock.....	90
Figure 27 - DDR DCM Implementation	92
Figure 28 - PPC External Clock.....	93
Figure 29 - REFCLK/BREFCLK Selection Logic	95
Figure 30 - Reset Topology Block Diagram	97
Figure 31 - SSRAM Connection	99
Figure 32 - SSRAM Flow-through	100
Figure 33 - SSRAM Pipeline.....	101
Figure 34 - SSRAM ZBT Flow-through.....	101
Figure 35 - SSRAM ZBT Pipeline.....	101
Figure 36 - Syncburst and ZBT SSRAM Timing.....	102
Figure 37 - Clock Level Translation	103
Figure 38 - DDR SDRAM Connection	120
Figure 39 - SSTL2 Class 1 Termination.....	121
Figure 40 - SSTL2 Class 2 Termination.....	121
Figure 41 - DDR VTT Termination Regulator	122
Figure 42 - RocketIO Block Diagram.....	144
Figure 43 - CPU Debug Connector	148
Figure 44 - Combined Trace/Debug Connector Pinout.....	150
Figure 45 - VirtexII Pro PCI VCCO Regulator	155
Figure 46 - PCI Edge Connector.....	156
Figure 47 - M66EN and PCIXCAP Jumper.....	162
Figure 48 - ATX Power Supply.....	164
Figure 49 - External Power Connection.....	165
Figure 50 - Test Header.....	168
Figure 51 - Test Header Pin Numbering.....	168
Figure 52 - DN3000K10SD Daughter Card Block Diagram.....	169
Figure 53 - DN3000K10S Daughter Card	170

List of Tables

Table 1 – Jumper Description	10
Table 2: S1 Dipswitch Configuration Settings.....	60
Table 3: HyperTerminal Main Menu Options.....	61
Table 4: HyperTerminal Interactive Configuration Menu Options.....	63
Table 5 - FPGA Configuration Modes	73
Table 6 - FPGA configuration file sizes	74
Table 7 - Connection between Configuration FPGA/MCU	75
Table 8 - FPGA JTAG connection to Configuration FPGA	77
Table 9 - Clocking inputs to the FPGA's.....	78
Table 10 - Clock Source Signals	81
Table 11 - RoboClock Configuration Signals	84
Table 12 - Connection between FPGA and External PPC Oscillator	90
Table 13 - Connection between FPGA's and DDR PLL Clock Drivers	92
Table 14 - Connection between FPGA and External PPC Oscillator	93
Table 15 - Connections between FPGA's and Rocket IO Clock Synthesizers.....	95
Table 16 - PPC Reset	98
Table 17 - Connection between FPGA and SRAM	103
Table 18 - Connection between FPGA's and DDR SDRAM's	123
Table 19 - Connections between FPGA and SMA Connectors.....	144
Table 20 - RocketIO Performance	146
Table 21 - CPU Debug connection to FPGA	148
Table 22 - Combined CPU Trace/Debug connection to FPGA	150
Table 23 - CPLD LED's.....	152
Table 24 - MCU LED's	153
Table 25 – FPGA A GPIO LED's.....	153
Table 26 - PCI to FPGA Connections	156
Table 27 - Present Signal Definition	161
Table 28 - M66EN and PCIXCAP Encoding.....	162
Table 29 – Voltage Indicators	165
Table 30 - External Power Connections.....	172
Table 31 - Connection between FPGA and the Daughter Card Headers	174
Table 32: bar_write_byte Arguments.....	213
Table 33: bar_write_word Arguments	214
Table 34: bar_write_dword Arguments.....	215
Table 35: bar_read_byte Arguments	216
Table 36: bar_read_word Arguments	217
Table 37: bar_read_dword Arguments	218
Table 38: dma_buffer_allocate Arguments.....	219
Table 39: dma_buffer_free Arguments.....	220
Table 40: dma_write_dword Arguments	221
Table 41: dma_read_dword Arguments.....	222
Table 42: pci_rdwrr Arguments	223
Table 43: DeviceIoControl Arguments	225

About This Manual

This User Guide accompanies the DN6000K10PCI LOGIC Emulation Board. For specific information regarding the Virtex-II Pro parts, please reference the datasheet.

1 Manual Contents

This manual contains the following chapters:

Chapter 1, “About This Manual”, how to use this manual, and additional resources.

Chapter 2, “Getting Started”, contains information on the contents of the LOGIC Emulation Kit.

Chapter 3, “Introduction to USB Controller Software”, description of the USB application software, used for configuration.

Chapter 4, “Introduction to the Virtex-II Pro and ISE”, an overview of the Virtex-II platform and the software features.

Chapter 5, “Introduction to the Software Tools”, information regarding the reference design and test software.

Chapter 6, “Programming/Configuring the Hardware”, step-by-step information on programming and configuring the hardware.

Chapter 7, “Board Hardware”, detailed description of board hardware.

2 Additional Resources

For additional information, go to <http://www.dinigroup.com/>. The following table lists some of the resources you can access from this website. You can also directly access these resources using the provided URLs.

Resource	Description/URL
User Manual	This is the main source of technical information. The manual should contain most of the answers to your questions
Dini Group Web Site	The web page will contain the latest manual, application notes, FAQ, articles, and any device errata and manual addenda. Please visit and bookmark: http://www.dinigroup.com
Data Book	Pages from The Programmable Logic Data Book, which contains device-specific information on Xilinx device characteristics, including readback, boundary scan, configuration, length count, and debugging http://direct.xilinx.com/bvdocs/publications/ds083.pdf
E-Mail	You may direct questions and feedback to the Dini Group using this e-mail address: support@dinigroup.com
Phone Support	Call us at 858.454.3419 during the hours of 8:00am to 5:00pm Pacific Time.
FAQ	The download section of the web page contains a document called DN6000K10PCI Frequently Asked Questions (FAQ) . This document is periodically updated with information that may not be in the User's Manual.

3 Conventions

This document uses the following conventions. An example illustrates each convention.

3.1 Typographical

The following typographical conventions are used in this document:

Convention	Meaning or Use	Example
Courier font	Messages, prompts, and program files that the system displays	<code>speed grade: - 100</code>

Convention	Meaning or Use	Example
Courier bold	Literal commands that you enter in a syntactical statement	ngdbuild <i>design_name</i>
Garamond bold	Commands that you select from a menu	File → Open
	Keyboard shortcuts	Ctrl+C
<i>Italic font</i>	Variables in a syntax statement for which you must supply values	ngdbuild <i>design_name</i>
	References to other manuals	See the <i>Development System Reference Guide</i> for more information.
	Emphasis in text	If a wire is drawn so that it overlaps the pin of a symbol, the two nets are <i>not</i> connected.
Braces []	An optional entry or parameter. However, in bus specifications, such as bus[7:0], they are required.	ngdbuild [<i>option_name</i>] <i>design_name</i>
Braces { }	A list of items from which you must choose one or more	lowpwr = {on off}
Vertical bar	Separates items in a list of choices	lowpwr = {on off}
Vertical ellipsis - - -	Repetitive material that has been omitted	IOB #1: Name = QOUT' IOB #2: Name = CLKIN' - -
Horizontal ellipsis . . .	Repetitive material that has been omitted	allow block <i>block_name</i> <i>loc1 loc2 ... locn;</i>
Prefix "0x" or suffix "h"	Indicates hexadecimal notation	Read from address 0x00110373, returned 4552494h
Letter "#" or "_n"	Signal is active low	INT# is active low fpga_inta_n is active low

3.2 Online Document

The following conventions are used in this document:

Convention	Meaning or Use	Example
Blue Text	Cross-reference link to a location in the current file or in another file in the current document	See the section “ Additional Resources ” for details. Refer to “ Title Formats ” in Chapter 1 for details.
Red Text	Cross-reference link to a location in another document	See Figure 2-5 in the <i>Virtex-II Pro Handbook</i>
Blue, underlined text	Hyperlink to a website (URL)	Go to http://www.xilinx.com for the latest datasheets.

4 Relevant Information

Information about PCI can be obtained from the following sources:

Reference the PCI Special Interest Group for the latest in PCI Express Specifications:

PCI Special Interest Group <http://www.pcisig.com>

5440 SW Westgate Dr, #217

Portland, OR 97221, USA

Phone: 503-291-2569

Fax: 503-297-1090

Other recommended specifications include:

PCI Industrial Computer Manufacturers Group (PICMG) <http://picmg.org>

401 Edgewater Place, Suite 600

Wakefield, MA 01880, USA

Phone: 781-224-1100

Fax: 781-224-1239

Suggested reference books (available from Amazon):

ABOUT THIS MANUAL

Samir Palnitkar, *Verilog HDL: A Guide to Digital Design and Synthesis*, ISBN: 0-13-451675-3

Sundar Rajan, *Essential VHDL: RTL Synthesis Done Right*

Edwin Breecher, *The IQ Booster: Improve Your IQ Performance Dramatically*

Getting Started

Congratulations on your purchase of the DN6000K10PCI LOGIC Emulation Board! You can begin by installing the software, or by powering on your DN6000K10PCI. If you wish to begin installation, please follow the installation instructions. The remainder of this chapter describes the contents of the box and how to start using the DN6000K10PCI LOGIC Emulation Board.

1 Precaution

The DN6000K10PCI is sensitive to static electricity, so treat the PCB accordingly. The target markets for this product are engineers that are familiar with FPGA's and circuit boards, so a lecture in ESD really isn't appropriate (and wouldn't be read anyway). However, the following web page has an excellent tutorial on the "Fundamentals of ESD" for those of you who are new to ESD sensitive products:

<http://www.esda.org/basics/part1.cfm>

The DN6000K10PCI has been factory tested and pre-programmed to ensure correct operation. You do not need to alter any jumpers or program anything to see the board work. A reference design is included on the enclosed CD. Please verify that the board is in working order by following the steps below:

2 The DN6000K10PCI LOGIC Emulation Kit

The DN6000K10PCI LOGIC Emulation Kit provides a complete development platform for designing and verifying applications based on the Xilinx, Virtex-II Pro FPGA family. The DN6000k10PCI is stand-alone or hosted via a USB interface. The DN6000K10PCI enables designers to implement embedded processor based applications with extreme flexibility using IP cores and customized modules. The

Virtex-II Pro FPGA with its integrated PowerPC processor and powerful Rocket I/O, Multi-Gigabit Transceivers (MGT) make it possible to develop highly flexible and high-speed serial transceiver applications.

The DN6000K10PCI, in its standard configuration, includes a high speed USB interface, a SmartMedia interface for configuration, 16M x 16 DDR SDRAM (x8), 4M x 16 FLASH (x5), RS232 ports (x4 multiplexed) and a RS232 monitor port. There are 6 low skew clock sources that are distributed to the FPGAs and the test header. A 200-pin test header allows for connection to individual FPGA's IO banks, using a custom daughter card. [Figure 1](#) - shows the DN6000K10PCI Logic Emulation Board.



Figure 1 - DN6000K10PCI LOGIC Emulation Board

The DN6000K10PCI LOGIC Emulation Kit includes the following:

- ✓ DN6000K10PCI development board (2VP70 or 2VP100 in the FF1704 package) Note: Specific speed grade parts required for various RocketIO/Power PC operating speeds, refer to Xilinx datasheet).
- ✓ 32MB SmartMedia Card, with reference design and main.txt
- ✓ 32MB SmartMedia Card, for customer use (blank)
- ✓ FlashPath Adapter to copy bit files to the SmartMedia Card(s)
- ✓ RS232 Serial cable, female to female (6ft)
- ✓ IDC 10-pin to DB 9-pin adaptor cable

- ✓ Jumpers 0.1”(x10)
- ✓ Documentation/Reference CD

Optional items that support development efforts (not provided):

- ✓ Xilinx ISE software
- ✓ JTAG cable
- ✓ Daughter Card

3 Installation Instructions

3.1 Jumper Setup

[Figure 2](#) indicates the factory jumper configuration of the DN6000K10PCI.

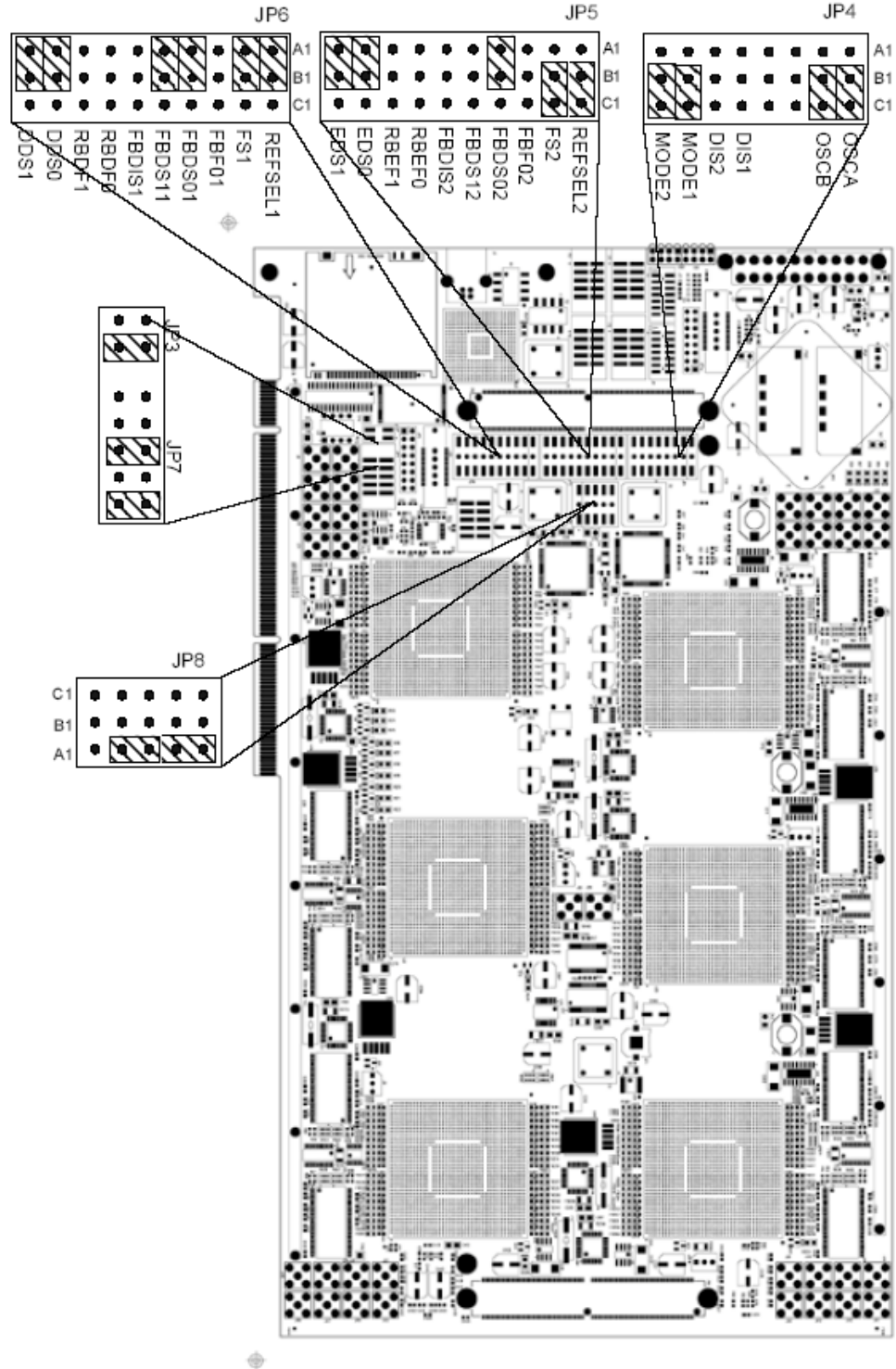


Figure 2 - Default Jumper Setup

3.2 Jumper Description

Table 1 – describes the functionality of the installed jumpers on the DN6000K10PCI.

Table 1 – Jumper Description

Jumper Installed	Signal Name	Description
JP8.A4-A3	PLL2BNC	CFPGA_CLKOUT connected to RoboClock #2 (U57). CFPGA_CLKOUT is an output clock from the Configuration FPGA. This connection causes 48MHz to output on all BCLK signals, which is used in the reference design for communication between the Configuration FPGA and VirtexII Pro FPGAs.
JP8.A5-B5	CLOCKB	Oscillator (X8 – 33.33MHz) connected to RoboClock #1 (U56).
JP6.A1-C1	ROBO1_REFSEL	ROBOCLOCK #1, Reference Select Input: The REFSEL input controls how the reference input is configured. When LOW, it will use the REFA pair (PLL1A) as the reference input. When HIGH, it will use the REFB pair (PLL1BC, PLL1BNC) as the reference input. This input has an internal pull-down.
JP6.A2-C2	ROBO1_FS	ROBOCLOCK #1, Frequency Select: This input must be set according to the nominal frequency (fNOM). Refer to Table 1 in the datasheet.
JP6.A4-B4	ROBO1_FBDS0	ROBOCLOCK #1, Feedback Divider Function Select: These inputs determine the function of the QFA0 and QFA1 outputs. Refer to Table 4 in the datasheet.
JP6.A9-B9	ROBO1_DS0	ROBOCLOCK #1, Output Divider Function Select: Controls the divider function of all banks (ACLKx) of outputs. Refer to Table 4 in the datasheet.
JP6.A10-B10	ROBO1_DS1	ROBOCLOCK #1, Output Divider Function Select: Controls the divider function of all banks (ACLKx) of outputs. Refer to Table 4 in the datasheet.

Jumper Installed	Signal Name	Description
		RoboClock #2 (U57)
JP5.A1-B1	ROBO2_REFSEL	ROBOCLOCK #2, Reference Select Input: The REFSEL input controls how the reference input is configured. When LOW, it will use the REFA pair (PLL1A) as the reference input. When HIGH, it will use the REFB pair (PLL1BC, PLL1BNC) as the reference input. This input has an internal pull-down.
JP5.A4-B4	ROBO2_FBDS0	ROBOCLOCK #2, Feedback Divider Function Select: These inputs determine the function of the QFA0 and QFA1 outputs. Refer to Table 4 in the datasheet.
JP5.A5-B5	ROBO2_FBDS1	ROBOCLOCK #2, Feedback Divider Function Select: These inputs determine the function of the QFA0 and QFA1 outputs. Refer to Table 4 in the datasheet.
JP5.A9-B9	ROBO2_DS0	ROBOCLOCK #2, Output Divider Function Select: Controls the divider function of all banks (BCLKx) of outputs. Refer to Table 4 in the datasheet.
JP5.A10-B10	ROBO2_DS1	ROBOCLOCK #2, Output Divider Function Select: Controls the divider function of all banks (BCLKx) of outputs. Refer to Table 4 in the datasheet.
JP4.B1-C1	OSCA	Enable for Oscillator A (X9)
JP4.B2-BC2	OSCB	Enable for Oscillator B (X8)
JP4.B7-C7	ROBO1_MODE	ROBOCLOCK #1, Output Mode: This pin determines the clock outputs' disable state. When this input is HIGH, the clock outputs will disable to high-impedance (HI-Z). When this input is LOW, the clock outputs will disable to "HOLD-OFF" mode. When in MID, the device will enter factory test mode.
JP4.B8-C8	ROBO2_MODE	ROBOCLOCK #2, Output Mode: This pin determines the clock outputs' disable state. When this input is HIGH, the clock outputs

Jumper Installed	Signal Name	Description
		will disable to high-impedance (HI-Z). When this input is LOW, the clock outputs will disable to “HOLD-OFF” mode. When in MID, the device will enter factory test mode.

3.3 Switch Setup and Description

Switch	Default Position	Signal Name	Description
S1.1	ON	FPGA_MSEL0	FPGA MSEL[0] – used to set configuration mode for all VirtexII Pro FPGAs
S1.2	OFF	FPGA_MSEL1	FPGA MSEL[1] – used to set configuration mode for all VirtexII Pro FPGAs
S1.3	OFF	FPGA_MSEL2	FPGA MSEL[2] – used to set configuration mode for all VirtexII Pro FPGAs
S1.4	OFF	DP_SW3	Not used
S3.1	ON	CFPGA_MSEL0	Configuration FPGA MSEL[0]
S3.2	ON	CFPGA_MSEL1	Configuration FPGA MSEL[1]
S3.3	ON	CFPGA_MSEL2	Configuration FPGA MSEL[2]
S3.4	ON	Not Connected	N/A

3.4 Oscillator Setup

The DN6000k10PCI is shipped from the factory with a 33.33MHz in X2, 14.31818 in X3, and 100MHz in X4. If the Roboclock jumpers are set to their default locations then ACLKx will be 133.33MHz and BCLKx will be 48MHz.

3.5 PPC RS232 Port Setup

There are 4 RS232 ports that are shared with the 6 VirtexII FPGAs. These ports are multiplexed by the Configuration FPGA and can be changed via the MCU Main Menu (see [Switch 4](#) on S2 tells MCU how to boot

- If the 4th switch position is ON then the MCU boot sequence will behave in the following manner:
 - (1) If the USB cable is plugged in when the DN6000k10 is powered-on/reset the MCU boots from the EEPROM (U8) and waits for USBController applicatin to send commands. In this case, the MCU FLASH firmware stored in U6 can be updated. In this state the MCU has limited USB functionality and cannot configure the FPGAs via USB/SmartMedia or perform many of the other USB GUI functions.
 - (2) If the USB cable is NOT plugged in when the DN6000k10 is powered-on/reset the MCU first boots from the EEPROM (U8) and then automatically boots from the MCU FLASH (U6). In this case, the MCU FLASH can NOT be updated.
- If the 4th switch position is OFF then the MCU will always boot from the MCU FLASH (U6) regardless of whether the USB cable is plugged in or not. When the MCU has booted from the FLASH it has full USB and FPGA configuration functionality. This is the default factory setup as of 1/1/05. Please note you can NOT update the MCU FLASH in the switch position.

Configuring HyperTerminal). The default setup is:

Port1 (P1): FPGA A

Port2 (P4): FPGA F

Port3 (P6): FPGA D

Port4 (P7): FPGA C

3.6 Powering ON the DN6000K10PCI

This section describes what is necessary to power-up the DN6000K10PCI

1. Install the SmartMedia card containing reference design into the DN6000K10PCI.
2. If switch position 4 on S1 is OFF then the MCU will automatically boot from the flash, and try to configure the FPGAs via the SmartMedia card (please see [Creating Configuration File “main.txt”](#) for information on setting up the files on the SmartMedia card). If the switch position 4 on S1 is ON then the MCU

will wait for USB commands and will not be able to configure the FPGAs until the USB application (on the product CD in “Source Code\USBController\USBController.exe”) is opened.

3. You can hook up the MCU RS232 port P7 to see messages during FPGA configuration (see [Configuring HyperTerminal](#) for more details).
4. Turn on the power.

WARNING: Do not use a separate ATX power supply with the board when it is plugged into a PCI slot! Use one of the 4-pin power connectors from the host machine.

4 Playing with your DN6000k10PCI via AETEST

At this point, the DN6000k10PCI should be powered on. All present FPGAs should be programmed with the reference design bit files provided by The Dini Group.

1. Power off the DN6000k10PCI.
2. Install the DN6000k10PCI into a PCI slot. Make sure the proper Smartmedia card is installed, with the PCI reference design on it.
3. Plug in an ATX power cable into the ATX port on the DN6000k10PCI.
4. Power ON the test PC and allow booting in DOS mode.

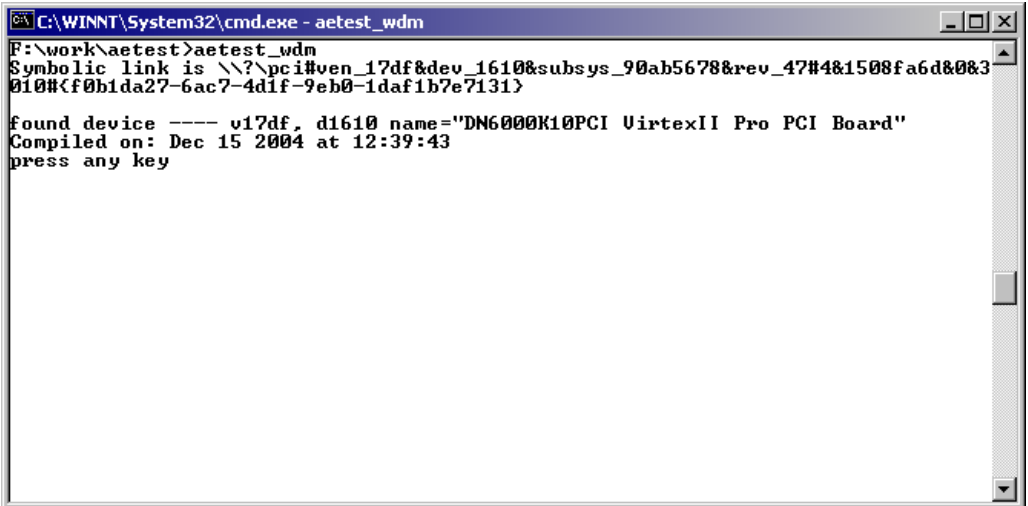
At this point, the DN6000k10PCI should be powered on with the PC booted in DOS mode. The FPGA should also be programmed with the PCI reference design supplied

Note: The FPGA programming will commence as soon as the DN6000k10PCI is powered on if the SmartMedia card contains the necessary configuration file and bit files. In general, the FPGA will be programmed prior to the PCI devices being configured. However, some computers have a “FastBoot” or “QuickBoot” feature which speeds up the booting process of the PC. These features are incompatible with the FPGA programming sequence of the DN6000k10PCI as the FPGA may not be configured prior to PCI bus activity. As a result, the DN6000k10PCI will not be recognized by the computer.

Workaround: If the computer has a “FastBoot” or “QuickBoot” (or similar) feature, it should be disabled. Otherwise, a soft-reset should be performed (by simultaneously pressing the CTRL-ALT-DELETE keys) after the computer has completed the Power-On Self Test (POST). This will allow the DN6000k10PCI enough time to configure the FPGA so that the computer will recognize the DN6000k10PCI device.

by The Dini Group. The ASIC Emulator Test Utility (AETEST) can now be used in DOS to verify the functionality of the DN6000k10PCI.

1. If the AETEST utility is not yet installed, refer to Appendix B for installation instructions.
2. Run the AETEST utility appropriate for the Operating System.
 - “AETESTDJ.EXE” for Windows 95/98/ME using DPPI
 - “AETEST98.EXE” for Windows 98/ME using VxD driver
 - “AETEST_WDM.EXE” for Windows 2000/XP
3. The AETEST utility should now recognize the DN6000k10PCI with the DEVICE_ID of 0x1600 and its VENDOR_ID of 0x17DF.



```
C:\WINNT\System32\cmd.exe - aetest_wdm
F:\work\etest>aetest_wdm
Symbolic link is \\?\pci#ven_17df&dev_1610&subsys_90ab5678&rev_47#4&1508fa6d&0&3
010#<f0b1da27-6ac7-4dif-9eb0-1daf1b7e7131>
found device ---- v17df, d1610 name="DN6000K10PCI UirtexII Pro PCI Board"
Compiled on: Dec 15 2004 at 12:39:43
press any key
```

4. Follow the on-screen instructions until the Main Menu is displayed.

```

C:\WINNT\System32\cmd.exe - aetest_wdm

---- ASIC Emulator PCI Controller Driver ---- v6
Compiled on: Dec 15 2004 at 12:39:38

P> PCI Menu
M> Memory Menu
Q> Quit

---- PCI BASE ADDRESS ----
0 : fc000000  1 : 00000000  2 : 00000000
3 : 00000000  4 : 00000000  5 : 00000000

Please select option: _

```

5. From the Main Menu, choose “Memory Menu”. The memory menu will now appear.

```

C:\WINNT\System32\cmd.exe - aetest_wdm

---- ASIC Emulator PCI Controller Driver ---- v6
Compiled on: Dec 15 2004 at 12:39:38

1) Write Dword <Same Address> 2) Read Dword <Same Address>
3) Write/Read Dword <Same Address>
4) BAR Memory Fill
5) BAR Memory Write
8) BAR Memory Display
p) bar memory range test
k) bar memory address/data bitwise test

c) test SRAM on FPGA A
d) test SRAM on FPGA B e) test DDR on FPGA B
h) test DDR on FPGA C
j) test DDR on FPGA D
l) test SRAM on FPGA E m) test DDR on FPGA E
n) test SRAM on FPGA F o) test DDR on FPGA F
x) Full Memory Test

M> Main Menu Q> Quit
---- PCI BASE ADDRESS ----
0 : fc000000  1 : 00000000  2 : 00000000
3 : 00000000  4 : 00000000  5 : 00000000

Please select option: _

```

6. The DN6000k10PCI features DDR SDRAM, SRAM, and Flash memory devices. The DN6000k10PCI specific memory tests are designed to exercise and verify the functionality of those features. Select one of the memory devices to be tested.

```

C:\WINNT\System32\cmd.exe - aetest_wdm
1> Write Dword <Same Address> 2> Read Dword <Same Address>
3> Write/Read Dword <Same Address>
4> BAR Memory Fill
5> BAR Memory Write
8> BAR Memory Display
p> bar memory range test
k> bar memory address/data bitwise test

c> test SRAM on FPGA A
d> test SRAM on FPGA B e> test DDR on FPGA B
h> test DDR on FPGA C
j> test DDR on FPGA D
l> test SRAM on FPGA E m> test DDR on FPGA E
n> test SRAM on FPGA F o> test DDR on FPGA F
x> Full Memory Test

M> Main Menu Q) Quit
==== PCI BASE ADDRESS ====
0 : fc000000 1 : 00000000 2 : 00000000
3 : 00000000 4 : 00000000 5 : 00000000

Please select option: h
Dword count? 0x10
Stop if an error occurs? <y or n>

```

7. The AETEST Test utility will now test the selected memory device using the memory controllers available in the PCI reference design. Press any key to exit the selected memory device test.

Congratulations! You have now programmed the DN6000k10PCI and successfully executed our AETEST utility to exercise various features of the DN6000k10PCI.

5 Playing with your DN6000k10PCI via the USB interface

At this point, the DN6000k10PCI should be powered on. All present FPGAs should be programmed with the reference design bit files provided by The Dini Group.

1. Hook up the USB cable to your DN6000k10PCI and your PC.
2. When you plug in the board and start windows the first time windows should detect the board and ask for a driver. Select "install from a list" -> select "search for the best driver in these locations". Select "include the location in the search" and browse to where the INF file is located (on the product CD in "Source Code\AETEST_USB\driver\win_wdm\") ->select "finish"
3. If the driver was installed successfully you should see the following device in the USB section of the device manager: "DiniGroup DN6000k10PCI FLASH boot".
4. You can now run the USB application found on the product CD in "Source Code\USBController\USBController.exe".

5. Please see [Exploring the Software Tools](#) for more information on the USBController application.

6 Playing with your DN6000K10PCI via the PPC's

At this point, the DN6000K10PCI should be powered on. All present FPGA's should be programmed with the reference design bit files supplied by The Dini Group.

6. Hook up the PPC RS232 port 1 (P1). All PPC RS232 ports run at 19200 bps.
7. Press '1' on the MCU menu to reconfigure all FPGA's. When configuration is complete the following text will be displayed on the PPC RS232 port:

```
*****
*****
** DN6000K10PCI ASIC DEVELOPMENT PLATFORM **
***** REFERENCE DESIGN SOFTWARE *****
*****
*****
```

FPGA_A:

Waiting for External Host Commands

Press Any Key To Enter Local User Menu

8. At this point tests may be run from the MCU menu. Text will appear on the PPC RS232 port as tests from the MCU menu are run on the associated FPGA (At this point the PPC port is connected to FPGA A).
9. Press a key on the PPC RS232 port to display the PPC test menu. See the section [Using the Reference Design](#) in [Chapter 4: Introduction to the Reference Design](#) for more information.

Congratulations! You have now programmed the DN6000K10PCI and successfully executed our utility to exercise various features of the board. All of the source code for the embedded PowerPC utility is included on the CD for reference. The FPGA design, written in Verilog, can also be found on the CD and used as a basis for a new design.

Introduction to USB Controller Software

This chapter introduces the DN6000K10PCI USB Graphical User Interface, including information required to configure the FPGA's via USB.

1 Exploring the Software Tools

1.1 USBController

USBController application is used to communicate with the DN6000k10PCI.

All USBController source code is included on the CD-ROM shipped with the DN6000k10PCI. The USBController can be installed on Windows 98/ME/2000/XP. There is a command line version called AETEST_USB that can be installed on Linux and Solaris.

There are 2 versions of the USBController application:

- o USBControllerUpdate.exe – Allows the user to update the MCU Flash
- o USBController.exe – Does not allow the user to update the MCU Flash

The MCU Flash would need to be updated if the user wants to modify any of the MCU firmware or if there is an updated FLASH provided by The Dini Group.

The USBController Application contains the following functionality:

- o Configure FPGA(s) over USB
- o Verify Configuration Status
- o Configure FPGAs via Smartmedia card

- o Clear FPGA(s)
- o Reset FPGA(s)
- o Set RocketIO CLK Frequency
- o Turn FPGA Fan(s) Off/On
- o Retrieve MCU/Spartan version
- o Update MCU FLASH firmware (only available with USBControllerUpdate)

The following features are only available when The Dini Group Reference design bit files are loaded:

- o Read/Write to FPGA(s) – see Appendix A for address maps
- o Test DDRs/FLASH/Reigsters/Interconnect

1.1.1 Getting Started with USBController

Once USBController is installed and the DN6000k10PCI is powered on and the USB cable is plugged in, the user can open USBController. The USBController application should immediately find the DN6000k10PCI. If USBController does not find the DN6000k10PCI, the user will get the following alert (see Figure 3)

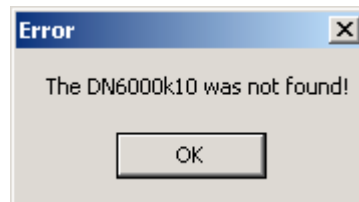


Figure 3 DN6000k10PCI Not Found

1.1.2 Basic Menu Operations

If the USBController finds the DN6000k10PCI and the USB cable was plugged into the PC before power was turned on to the DN6000k10PCI you will see the following screen:

Figure 4: Booting from FLASH

If the USB Cable was plugged into the DN6000k10PCI after it powered on you will see the following screen:

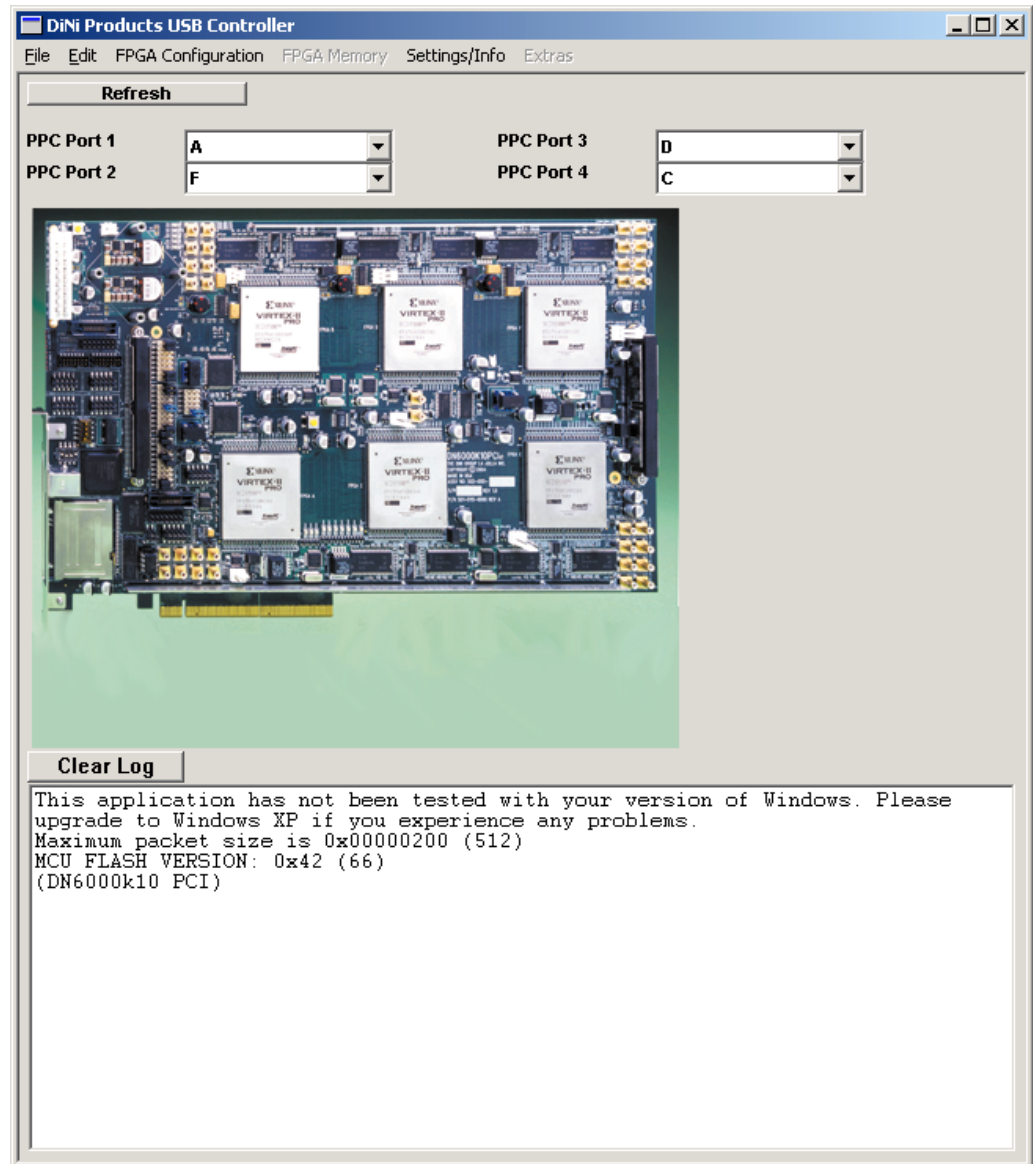


Figure 5: Main USBController Screen

1.1.3 Enable/Disable USB to FPGA Communication

Disable USB->FPGA Com. ?

This button allows you to disable the USB to FPGA communication via the Spartan II. When the USB interface is used, the Spartan II will drive main bus (MB) pins 0-39 in order to provide USB communication to the FPGAs. This makes main bus pins 0-39 unusable for any other purpose. If your design requires the use of these pins it is necessary to disable USB to FPGA communication, which will cause the Spartan II to cease driving these pins and release them for other purposes.

Note: In order to run our reference design, USB to FPGA communication **must** be enabled.

Note: USB to FPGA communication is disabled by default.

1.1.4 File Menu

The File Menu has the following 2 options:

- (1) Open – opens a file with the selected text editor (notepad by default). To change the text editor see Settings/Info Menu section
- (2) Exit – Closes the USBController application

1.1.5 Edit Menu

The Edit Menu performs the basic edit commands on the command log in the bottom half of the USBController window. The Find option is not currently supported.

1.1.6 FPGA Configuration Menu

The FPGA Configuration Menu has the following options:

- (1) Refresh Configuration Status – Queries to see which FPGA(s) are configured and update the GREEN LEDS in DN6000k10PCI picture
- (2) Configure via USB (individually) – After selecting this option a window will pop and ask which FPGA you want to configure and then what bitfile you want to configure the selected FPGA with. The status of the FPGA configuration will detailed in the log window and the DN6000k10PCI will be updated after the bitfile has been transferred.
- (3) Configure via USB using file – This option allows the user to configure more than one FPGA over USB at a time. To use this option you must create a setup file that contains information on which FPGA(s) should be configured and what bitfiles should be used for each FPGA. The file should be in the following format, the first character of each line represents which FPGA you want configured (a-f or A-F), this letter should be followed by a colon and then the path to the bitfile to use for this FPGA. The path to the bitfile is realative to the directory where this setup file is, or you can use the full path. Below is an example of an accepted setup file:

A: fpga_a.bit

B: fpga_b.bit

C: fpga_c.it

- (4) Configure via SmartMedia Card – This option allows the user to use a SmartMedia card to configure the FPGAs. Please see section [Creating Configuration File “main.txt”](#) for information on what files should be on the SmartMedia card to use this option.
- (5) Clear All FPGAs – This option will clear all FPGAs of configuration.
- (6) Reset – This option sends an active low reset (active for approx. 20ns) to all FPGAs on the signal called FPGA_GRSTn which is connected to the following I/O pins:

FPGA A: AA12

FPGA B: M22

FPGA C: E20

FPGA D: M16

FPGA E: E20

FPGA F: AV7

1.1.7 FPGA MemoryMenu

The FPGA Memory Menu has the following options

- (1) Write DWORD(s) – Writes DWORD(s) to memory with a specified starting address, and number of DWORD(s) to write. Also, the user can specify what to write. (address value, inverse of address, or a user inputted value) Additionally, enabling verbose mode will allow the user to see what has been written, to what address. Please note that all addresses must be entered as 8-digit hexadecimal.
- (2) Read DWORD(s) – Reads DWORD(s) from memory with a specified starting address, and number of DWORD(s) to read. The “Values used to test memory” options are non-functional for this dialog. Verbose mode will print what is read from what address.
- (3) Write and Read DWORD(s) – this combines the previous two items. It first writes the DWORD(s) to a given address range, and then reads back those addresses. The (values used to test memory” will determine what is written to each address.

- (4) Test Address Space – this will write the specified value to a given address range, read it back, and check the results for errors. Note: some memory addresses cannot be written to, and will return errors. Please check the FPGA memory maps in Appendix A for clarification.
- (5) Display Address Space – coming soon!
- (6) Test DDR (through PPC's) – tests an FPGA's DDR by using the PPC built into each FPGA.
- (7) Test FLASH (through PPC's) – tests an FPGA's flash by using the PPC built into each FPGA.
- (8) Test SRAM (through PPC's) – tests an FPGA's SRAM by using the PPC built into each FPGA.
- (9) Test Internal Registers – coming soon!
- (10) Test Interconnect – coming soon!
- (11) Test ALL (through PPC's) – tests an FPGA's DDR, flash, and SRAM by using the PPC built into each FPGA.
- (12) Display Memory Map – coming soon!

1.1.8 Settings/Info Menu

The Settings/Info Menu has the following options

- (1) Set FPGA RocketIO CLK Frequency – When the DN6000k10PCI is first powered up the RocketIO CLK inputs to the FPGAs are inactive. The RocketIO CLK Inputs are connected to the following FPGA Differential CLK inputs on all FPGAs: F21/G21 and AT21/AU21. This menu option allows the user to specify what frequency the RocketIO CLKs should be set at for each FPGA. The supported frequency range is 31.25MHz – 700MHz. After selecting this option, a pop-up window will ask which FPGA's RocketIO Frequency you want to set (or you can choose to set all to the same frequency), and then what frequency you want. Check the log window to verify what frequency the CLKs were actually set at.
- (2) Change Text Editor – This options allows the user to select a text editor to use (the default editor is notepad).
- (3) FPGA Stuffing Information – This option will display the type of FPGAs that are stuffed on the DN6000k10PCI.

- (4) Turn FPGA Fans On/Off – This option will either turn the FPGA fans on or off.
- (5) MCU Firmware Version – This option will display the MCU Firmware version in the log window.
- (6) BOARD/SPARTAN Version – This option will display the Board Version along with the Spartan (Config Fpga) Version.

1.2 PCI AETEST Application

AETEST utility program is used primarily to test and verify the functionality of the DN6000K10PCI Logic Emulation board.

All AETEST source code is included on the CD-ROM shipped with your DN6000K10PCI Logic Emulation kit. AETEST can be installed on a variety of operating systems, including:

- DOS and Windows 95/98/ME using DPMI (DOS Protected Mode Interface)
- Windows 98/ME using a VxD driver
- Windows 2000/XP (Windows WDM)
- Windows NT
- Linux
- Solaris

The AETEST utility program contains the following tests:

- PCI Test
- Memory Tests (SRAM & DDR)
- FLASH Test
- Daughter Card Test (with or without cables)
- BAR Memory Range Tests

AETEST also provides the user with the following abilities:

- Recognize the DN6000K10S

- Read FPGA F Revision
- Display Vendor and Device ID
- Set PCI Device and Function Number
- Display all configured PCI devices
- Various loops for PCI device-function and ID numbers
- Write and Read Configuration DWORD
- Write DWORD, Read DWORD and Write/Read DWORD (Same Address)
- BAR Memory Fill, Write and Display
- Configure/Save BAR's from/to a file

More information coming soon.

Introduction to Virtex-II Pro and ISE

1 Virtex-II Pro

The Virtex-II Pro FPGA solution is the most technically sophisticated silicon and software product development in the history of the programmable logic industry. The goal was to revolutionize system architecture “from the ground up.” To achieve that objective, the best circuit engineers and system architects from IBM, Mindspeed, and Xilinx co developed the world's most advanced FPGA silicon product. Leading teams from top embedded systems companies worked together with Xilinx software teams to develop the systems software and IP solutions that enabled new system architecture paradigm.

The result is the first FPGA solution capable of implementing high performance system-on-a-chip designs previously the exclusive domain of custom ASICs, yet with the flexibility and low development cost of programmable logic. The Virtex-II Pro family marks the first paradigm change from programmable logic to programmable systems, with profound implications for leading-edge system architectures in networking applications, deeply embedded systems, and digital signal processing systems. It allows custom user-defined system architectures to be synthesized, next-generation connectivity standards to be seamlessly bridged, and complex hardware and software systems to be co-developed rapidly with in-system debug at system speeds. Together, these capabilities usher in the next programmable logic revolution.

1.1 Summary of Virtex-II Pro Features

The Virtex-II Pro has an impressive collection of both programmable logic and hard IP that has historically been the domain of the ASICs.

- High-performance FPGA solution including:
 - Up to twenty-four RocketIO™ embedded multi-gigabit transceiver blocks (based on Mindspeed's SkyRail™ technology)

- Up to four IBM® PowerPC™ RISC processor blocks
- Based on Virtex™-II FPGA technology
 - Flexible logic resources, up to 125,136 Logic Cells
 - SRAM-based in-system configuration
 - Active Interconnect™ technology
 - SelectRAM™ memory hierarchy
 - Up to 556 Dedicated 18-bit x 18-bit multiplier blocks
 - High-performance clock management circuitry
 - SelectIO™-Ultra technology
 - Digitally Controlled Impedance (DCI) I/O

1.2 PowerPC™ 405 Core

- Embedded 300+ MHz Harvard architecture core
- Low power consumption: 0.9 mW/MHz
- Five-stage data path pipeline
- Hardware multiply/divide unit
- Thirty-two 32-bit general purpose registers
- 16 KB two-way set-associative instruction cache
- 16 KB two-way set-associative data cache
- Memory Management Unit (MMU)
 - 64-entry unified Translation Look-aside Buffers (TLB)
 - Variable page sizes (1 KB to 16 MB)
- Dedicated on-chip memory (OCM) interface
- Supports IBM CoreConnect™ bus architecture
- Debug and trace support
- Timer facilities

1.3 RocketIO 3.125 Gbps Transceivers

- Full-duplex serial transceiver (SERDES) capable of baud rates from 622 Mb/s to 3.125 Gb/s (please reference the Xilinx datasheet for speed grade limitations)
- 80 Gb/s duplex data rate (16 channels)

- Monolithic clock synthesis and clock recovery (CDR)
- Fibre Channel, Gigabit Ethernet, 10 Gb Attachment Unit Interface (XAUI), and
- Infiniband-compliant transceivers
- 8-, 16-, or 32-bit selectable internal FPGA interface
- 8B /10B encoder and decoder
- 50/75 on-chip selectable transmit and receive terminations
- Programmable comma detection
- Channel bonding support (two to sixteen channels)
- Rate matching via insertion/deletion characters
- Four levels of selectable pre-emphasis
- Five levels of output differential voltage
- Per-channel internal loopback modes
- 2.5V transceiver supply voltage

1.4 Virtex-II FPGA Fabric

Description of the Virtex-II Family fabric follows:

- SelectRAM memory hierarchy
 - Up to 10 Mb of True Dual-Port RAM in 18 Kb block SelectRAM resources
 - Up to 1.7 Mb of distributed SelectRAM resources
 - High-performance interfaces to external memory
- Arithmetic functions
 - Dedicated 18-bit x 18-bit multiplier blocks
 - Fast look-ahead carry logic chains
- Flexible logic resources
 - Up to 111,232 internal registers/latches with Clock Enable
 - Up to 111,232 look-up tables (LUTs) or cascadable variable (1 to 16 bits) shift registers
 - Wide multiplexers and wide-input function support
 - Horizontal cascade chain and Sum-of-Products support

- Internal 3-state busing
- High-performance clock management circuitry
 - Up to eight Digital Clock Manager (DCM) modules
 - Precise clock de-skew
 - Flexible frequency synthesis
 - High-resolution phase shifting
 - 16 global clock multiplexer buffers in all parts
- Active Interconnect technology
 - Fourth-generation segmented routing structure
 - Fast, predictable routing delay, independent of fanout
 - Deep sub-micron noise immunity benefits
- Select I/O-Ultra technology
 - Up to 852 user I/Os
 - Twenty two single-ended standards and five differential standards
 - Programmable LVTTTL and LVCMOS sink/source current (2 mA to 24 mA) per I/O
 - Digitally Controlled Impedance (DCI) I/O: on-chip termination resistors for single-ended I/O standards
 - PCI support(1)
 - Differential signaling
 - 840 Mb/s Low-Voltage Differential Signaling I/O (LVDS) with current mode drivers
 - Bus LVDS I/O
 - HyperTransport™ (LDT) I/O with current driver buffers
 - Built-in DDR input and output registers
 - Proprietary high-performance SelectLink technology for communications between Xilinx devices
 - High-bandwidth data path
 - Double Data Rate (DDR) link
 - Web-based HDL generation methodology
- SRAM-based in-system configuration
 - Fast SelectMAP™ configuration

- Triple Data Encryption Standard (DES) security option (bitstream encryption)
 - IEEE1532 support
 - Partial reconfiguration
 - Unlimited reprogrammability
 - Readback capability
- Supported by Xilinx Foundation™ and Alliance™ series development systems
 - Integrated VHDL and Verilog design flows
 - ChipScope™ Pro Integrated Logic Analyzer
- 0.13-μm, nine-layer copper process with 90 nm high-speed transistors
- 1.5V (VCCINT) core power supply, dedicated 2.5V VCCAUX auxiliary and VCCO power supplies
- IEEE 1149.1 compatible boundary-scan logic support
- Flip-Chip and Wire-Bond Ball Grid Array (BGA) packages in standard 1.00 mm pitch
- Each device 100% factory tested

2 Foundation ISE 6.1i

ISE Foundation is the industry's most complete programmable logic design environment. ISE Foundation includes the industry's most advanced timing driven implementation tools available for programmable logic design, along with design entry, synthesis and verification capabilities. With its ultra-fast runtimes, ProActive Timing Closure technologies, and seamless integration with the industry's most advanced verification products, ISE Foundation offers a great design environment for anyone looking for a complete programmable logic design solution.

2.1 Foundation Features

2.1.1 Design Entry

ISE greatly improves your “Time-to-Market”, productivity, and design quality with robust design entry features. ISE provides support for today's most popular methods for design capture including HDL and schematic entry, integration of IP cores as well as robust support for reuse of your own IP. ISE even includes technology called IP Builder, which allows you to capture your own IP and reuse it in other designs.

ISE's Architecture Wizards allow easy access to device features like the Digital Clock Manager and Multi-Gigabit I/O technology. ISE also includes a tool called PACE (Pinout Area Constraint Editor), which includes a front-end pin assignment editor, a

design hierarchy browser, and an area constraint editor. By using PACE, designers are able to observe and describe information regarding the connectivity and resource requirements of a design, resource layout of a target FPGA, and the mapping of the design onto the FPGA via location/area.

This rich mixture of design entry capabilities provides the easiest to use design environment available today for your logic design.

2.1.2 Synthesis

Synthesis is one of the most essential steps in your design methodology. It takes your conceptual Hardware Description Language (HDL) design definition and generates the logical or physical representation for the targeted silicon device. A state of the art synthesis engine is required to produce highly optimized results with a fast compile and turnaround time. To meet this requirement, the synthesis engine needs to be tightly integrated with the physical implementation tool and have the ability to proactively meet the design timing requirements by driving the placement in the physical device. In addition, cross probing between the physical design report and the HDL design code will further enhance the turnaround time.

Xilinx ISE provides the seamless integration with the leading synthesis engines from Mentor Graphics, Synopsys, and Synplicity. You can use the synthesis engine of your choice. In addition, ISE includes Xilinx proprietary synthesis technology, XST. You have options to use multiple synthesis engines to obtain the best-optimized result of your programmable logic design.

2.1.3 Implementation and Configuration

Programmable logic design implementation assigns the logic created during design entry and synthesis into specific physical resources of the target device.

The term “place and route” has historically been used to describe the implementation process for FPGA devices and “fitting” has been used for CPLDs. Implementation is followed by device configuration, where a bitstream is generated from the physical place and route information and downloaded into the target programmable logic device.

To ensure designers get their product to market quickly, Xilinx ISE software provides several key technologies required for design implementation:

- Ultra-fast runtimes enable multiple “turns” per day
- ProActive™ Timing Closure drives high-performance results
- Timing-driven place and route combined with “push-button” ease
- Incremental Design

- Macro Builder

2.1.4 Board Level Integration

Xilinx understands the critical issues such as complex board layout, signal integrity, high-speed bus interface, high-performance I/O bandwidth, and electromagnetic interference for system level designers.

To ease the system level designers' challenge, ISE provides support to all Xilinx leading FPGA technologies:

- System IO
- XCITE
- Digital clock management for system timing
- EMI control management for electromagnetic interference

To really help you ensure your programmable logic design works in context of your entire system, Xilinx provides complete pin configurations, packaging information, tips on signal integration, and various simulation models for your board level verification including:

- IBIS models
- HSPICE models
- STAMP models

3 Virtex-II Pro Developer's Kit

V2PDK is the Virtex-II Pro Developer's Kit, and is included to provide an existing framework of hardware and software code to explore the capabilities of the Virtex-II Pro, as well as a basis to build new systems.

A wide variety of software and hardware tools are used to build a Virtex-II Pro™ design. V2PDK The design flow is a tool chain methodology that exists to simplify the entire design process by providing integration between the tools and automating tasks. The main focus of the design flow is integrating the programs with each other to accomplish the system design.

The system design process can be loosely divided into the following tasks:

- Builds the software application

- Simulates the hardware description
- Simulates the hardware with the software application
- Simulates the hardware into the FPGA using the software application in on-chip memory
- Runs timing simulation
- Configures the bitstream for the FPGA

Introduction to the Reference Design

This chapter introduces the DN6000K10PCI Reference Design, including information on what the reference design does, how to build it from the source files, and how to modify it for another application.

1 Exploring the Reference Design

1.1 What is the Reference Design?

The reference design is a fully functional Virtex II Pro FPGA design capable of demonstrating most of the features available on the DN6000K10PCI. Features exercised in the reference design include:

- Access to the DDR SDRAM Modules At 133Mhz
- UART Communication
- FPGA Interconnect
- Interaction with the Configuration FPGA and MCU
- Use of Embedded PowerPC Processors
- Memory Mapped Access Between PPC And User Design
- Access to external LED's
- Communication via Rocket I/O Transceivers
- Instantiation of Daughter Card Test Headers

All source code for the reference design is included on the CD and may be used freely in customer development. Precompiled bit files for the most common stuffing

options are also included and should be used to verify board functionality before beginning development. A build utility, described in the section [Compiling The Reference Design](#), can be used to generate new bit files, or to generate bit files for less common configurations of the DN6000K10PCI.

1.2 Using the Reference Design

For information on preparing the board for running the reference design, see [Chapter 5: Programming / Configuring the Hardware](#). This section assumes that board has been set up with appropriate jumper settings and oscillators, code has been loaded for the Configuration FPGA and the MCU, and that the Reference Design has been loaded into at least FPGA A. **Note that when the board is shipped, all of these steps have already been completed- no modification to jumper settings, oscillators, Config FPGA code, or MCU code is required to use the Reference Design.**

The primary interface to the DN6000K10PCI Reference Design is through an RS232 Serial Port, connected to one of the four PPC RS232 headers, P1, P3, P4, and P2. For more information, see the section [PPC RS232 Port Setup](#) in [Chapter 2: Getting Started](#), and the section [Configuring HyperTerminal](#) in [Chapter 5: Programming / Configuring the Hardware](#). It is assumed at this point that a terminal emulator is connected to PPC Port1 (Header P1), running at 19200 bps.

Powering up the board will display the following text on the terminal:

```
*****
*****
** DN6000K10PCI ASIC DEVELOPMENT PLATFORM **
***** REFERENCE DESIGN SOFTWARE *****
*****
*****
```

FPGA_A:

Waiting for External Host Commands

Press Any Key To Enter Local User Menu

The various functions of the Reference Design may be controlled both from the MCU menu, described in the section [Description of Main Menu Options](#) in [Chapter 5](#), or from the PowerPC menu. In this example we will be using the PowerPC menu to exercise the functions of the Reference Design. When presented with the above text, the Reference Design is waiting for commands to be sent from the MCU. Press any key to stop waiting for MCU commands and get the following menu:

FPGA_A: MAIN MENU

- a) Run Full Test Suite
- b) Test Registers
- c) Test SRAM
- d) Test DDR
- e) Test Interconnect
- f) Write Memory Location
- g) Read Memory Location
- h) Display Memory in 8 DWORDS per Line Format
- i) Fill Memory with specified DWORD pattern
- j) Toggle Mem Owner: INTERNAL (User)
- k) Interconnect Test Menu

- q) Quit

Now tests can be run directly from the embedded PPC processor. The menu options are as follows:

- a. Run Full Test Suite: Runs options b,c,d, and e
- b. Test Registers: Runs read/write tests on local FPGA registers
- c. Test SRAM: Runs a full set of tests on the SRAM memory.
- d. Test DDR: Runs read/write tests on the DDR memories.
- e. Test Interconnect: Runs an inter-FPGA test on the physical interconnect.
- f. Write Memory Location: Allows writing to any PPC memory location
DDR_BASE = 0x80000000
SRAM_BASE = 0x90000000
REGISTER BASE = 0x98000000
- g. Read Memory Location: Allows reading any PPC memory location
- h. Display Memory... : Starting from any PPC address, lists DWORDs
- i. Fill Memory with specified DWORD pattern: Allows large chunks of memory to be filled with a known value.
- j. Toggle Mem Owner: Sets the Memory Arbiter to User (PPC) or Host
- k. Interconnect Test Menu: For interconnect debug (under construction)

Note that the full test suite takes about 5 minutes to run. To abort any test operation uses the PPC Reset Button (S4) to reset the design.

1.3 Compiling the Reference Design

This section deals with the source code to the Reference Design, which can be found on the CD-ROM. All file references are with respect to the root directory of the Reference Design source code (/source/FPGA). Files that are specific to the DN6000K10PCI design are found in the DN6000K10PCI subdirectory, whereas general application code is found in the common subdirectory.

1.3.1 The Xilinx Embedded Development Kit (EDK)

The Reference Design uses the Xilinx EDK to instantiate an embedded PowerPC Processor. The EDK project can be found at 'DN6000K10/PPC/system.xmp' and can be opened and modified with the Xilinx Embedded Development Kit software.

1.3.2 Synplicity Synplify

The Dini Group uses Synplicity's Synplify software to for design synthesis. The Synplicity projects for each of the 6 FPGA's on the DN6000K10PCI can be found at 'DN6000K10/synthesis/*.prj'. These projects have been compiled using Synplify Pro version 7.3.

1.3.3 Xilinx ISE

A sample Project Navigator project is located at 'DN6000K10/implement/fpga.npl'. For information on using Xilinx ISE, see the section [Foundation ISE 6.1i](#) in Chapter 3.

1.3.4 The Build Utility: Make.bat

The Build Utility is found at 'DN6000K10/build/make.bat'. This batch file is used to set system parameters to the desired configuration (i.e. VP70 vs. VP100, DDR2 stuffed or not stuffed, etc.), and to invoke all of the above tools from the command line. Instructions for invoking the batch file can be found by viewing the batch file with a text editor. Additional information about using the batch file to build the reference design is found below. Taking the reference design through all of the various tools for several FPGA's can be very tedious and time consuming- this batch file can do it all in one command!

The command line utility "Make.bat" is an MS-DOS batch file compatible with Windows 2000 and later operating systems. Make.bat should be run from the command line, with command line parameters. It should not be double clicked from the windows environment. A command prompt shortcut is provided in the same directory as Make.bat, and can be double clicked to open a command prompt window with the proper working directory.

Four main steps are involved in building the reference design. First the PowerPC netlist must be built using the EDK. The first time this is done it must be done from the EDK GUI, not from the command line. Open the EDK project (in PPC/system.xmp), and select Tools->make netlist. Once this has been done once, the Make.bat script can be used to build the netlist with the command Make ppc_netlist. The second step is to synthesize the design with Synplicity's Synplify Pro. The third

step in to place and route, or “implement” the design with the Xilinx ISE tools. The fourth and final step is to compile the PowerPC code and embed it in the bitfile. This fourth step is referred to by Xilinx as “updating” the bitfile. Hence this fourth step will be referred to as the “update” step.

The build script creates a directory called “out” and places its output files there. After the script completes you will find 3 files for each FPGA that was built. Fpga_*.bit is the file to be downloaded to the FPGA. The fpga_*_ui.bit and the fpga_*.bmm files are used by the Xilinx EDK in the “update” process to embed the PowerPC source code into the bitfile, creating the final bitfile.

All of the steps mentioned above can be performed with the build script. The following command line options are supported:

All	Synthesizes, implements, and updates for all 6 fpga's. Doesn't generate the PowerPC netlist.
*	Replace * with A, B, C, D, E, or F. Synthesizes, implements, and updates for the specified FPGA
synthesize_*	Replace * with a,b,c,d,e,f or all. Synthesizes the specified FPGA, or all FPGA's.
implement_*	Replace * with a,b,c,d,e,f, or all. Implements the specified FPGA, or all FPGA's.
update_*	Replace * with a,b,c,d,e,f or all. Updates the specified FPGA, or all FPGA's.
Clean	Deletes all intermediate tool-generated files. Leaves out directory intact.
clean_all	Deletes all generated files except those from the EDK
clean_ppc	Deletes all EDK netlist files
ppc_netlist	Rebuilds the EDK netlist. The netlist MUST previously have been build from the EDK user interface before it can be built from the command line.
make VP70	makes changes to synplicity and EDK project files, and UCF files to compile for VP70
make VP100	makes changes to synplicity and EDK project files, and UCF files to compile for VP100

make INCLUDE_DDR2	makes changes to synplicity projects, EDK source code, and UCF files to include DDR2
make EXCLUDE_DDR2	makes changes to synplicity projects, EDK source code, and UCF files to exclude DDR2
make DDR_32_MEG	makes changes to synplicity projects and EDK source code to set DDR size
make DDR_64_MEG	makes changes to synplicity projects and EDK source code to set DDR size
make DDR_128_MEG	makes changes to synplicity projects and EDK source code to set DDR size

The reference design must support any number of FPGA's in both VP70 and VP100 sizes. Compiler constants are used to include/exclude code, as well as to set appropriate parameters for the configuration being compiled for. Specifically, the user may want to include/exclude any memory device (DDR1, DDR2, SRAM), or may want to switch between the VP70 and VP100 part. There are four places where changes must be made to get the desired configuration:

- I. Synplicity synthesis project file
- II. UCF files in 'source/ucf'
- III. Xilinx EDK project file
- IV. Xilinx EDK processor source code files ('PPC/code/fpga_params/*.h')
- V. Setting up the build utility: "make.bat"

Note that the build utility runs the xilinx tools from the command line, so there are no Xilinx Project Navigator files to edit. If you choose to use the Project Navigator GUI, be very careful to have all the appropriate settings (ie 2vp70 vs 2vp100) The following sections explain what to change and what options the user has to accomplish these changes (Most are automated, some are not). It is highly recommended that everything be recompiled after making any of these changes, including the PPC netlist, the synplicity project, the Xilinx project, and the EDK source code. If everything is not updated properly unpredictable behavior will result. If you aren't sure, delete all tool generated files and start fresh. For information on the usage of the build tool (make.bat), see the top of the 'make.bat' file.

I. SYNPLICITY SYNTHESIS PROJECT FILE

In the 'synthesis' folder there are six project files, one for each FPGA. The line 'set_option -part XC2VP70' must be modified appropriately for the VP70 or VP100. This change, as well as changes to the parameters described below may be made

through the build utility (described below), through the synplicity GUI, or by hand. At the bottom of each file is a list of defined compiler constants that dictate what code is included and what code is excluded. The recognized constants are as follows:

EXTERNAL_DEFINES	Tells 'fpga.v' not to define it's own set of constants, but to use the ones defined externally (by synplicity).
FPGA_X	Tells 'fpga.v' which FPGA is being compiled. 'X' must be replaced with A,B,C,D,E,F,G,H, or I. Used to define the FPGA's ID number and name for communication with the host.
VP70/VP100	Tells 'fpga.v' which fpga is being targeted. Used by the interconnect test to disable bus lines that are no connect in the VP70 part.
INTERCON_MASTER	This must be defined for one and only one FPGA in the system. It includes the control code for the interconnect test- all other FPGA's are passive in the interconnect test. Any FPGA can be the master, but only one!
INCLUDE_SRAM EXCLUDE_SRAM	Includes/Excludes the SRAM controller code.
INCLUDE_DDR1 EXCLUDE_DDR1	Includes/Excludes the ddr1 controller code.
INCLUDE_DDR2 EXCLUDE_DDR2	Includes/Excludes the ddr2 controller code.
DDR_32_MEG DDR_64_MEG DDR_128_MEG	Defines the size of the DDR's, for address mapping.

The above parameters may be modified by hand, or by using the build utility with the following options:

make VP70	makes changes to synplicity and EDK project files, and UCF files to compile for VP70
make VP100	makes changes to synplicity and EDK project files, and UCF files to compile for VP100

make INCLUDE_DDR2	makes changes to synplicity projects, EDK source code, and UCF files to include DDR2
make EXCLUDE_DDR2	makes changes to synplicity projects, EDK source code, and UCF files to exclude DDR2
make DDR_32_MEG	makes changes to synplicity projects and EDK source code to set DDR size
make DDR_64_MEG	makes changes to synplicity projects and EDK source code to set DDR size
make DDR_128_MEG	makes changes to synplicity projects and EDK source code to set DDR size

II. UCF FILES

In 'source/ucf' is 6 UCF files, one for each FPGA. The UCF files must be modified to exclude any unused memory device (DDR1, DDR2, or FLASH). If any DDR or SRAM chip is to be excluded, simply comment out all associated lines in the UCF file by putting a '#' in front of the line. If DDR2 is to be excluded (it should always be excluded for the VP70), then the build utility may be used as shown below. Use 'make VP70' or make 'VP100' to include/exclude bus interconnect lines that are appropriate to that device.

Please note that the bus numbering in the files under 'source/ucf' *does not* match the schematic. We have included a set of UCF files that do match the schematic, under the directory 'source/ucf_busnum_1toN'. The UCF files under that directory, however, will not work with the reference design. You may use them for your own design if you wish. The difference between the two versions is that the standard UCF files (source/ucf) have busses with numbering starting from 0, while the UCF files matching the schematic (source/ucf_busnum_1toN) have busses with numbering starting at 1.

make INCLUDE_DDR2:	makes changes to synplicity projects, EDK source code, and UCF files to include DDR2
make EXCLUDE_DDR2	makes changes to synplicity projects, EDK source code, and UCF files to exclude DDR2
make VP70	comments out bus interconnect lines that are 'No Connect' in the VP70

	Connect' in the VP70
make VP100	uncomments bus interconnect lines that are 'No Connect' in the VP70, but exist in VP100

If excluding SRAM or DDR1, all changes must be made by hand (be sure to also make changes to the synplicity project file and the PPC source file 'PPC/code/fpga_params/*.h')

III. XILINX EDK PROJECT FILE

The Xilinx EDK Project file is found at 'PPC/system.xmp'. After making any changes to this file, be sure to select the 'clean all' option in the Xilinx EDK, so that all generated files will be remade with the new project settings. The only setting that should be changed in this file is the target device. This can be changed through the EDK GUI, using the build utility, or by hand. The device line looks like one of the following:

Device: xc2vp70

Device: xc2vp100

When changing between FPGA's the build utility can be used as follows:

make VP70	makes changes to synplicity and EDK project files and UCF files to compile for VP70
make VP100	makes changes to synplicity and EDK project files and UCF files to compile for VP100

IV. XILINX EDK PROCESSOR SOURCE CODE

The file 'PPC/code/fpga_params.h' defines the software parameters for the PowerPC part of the design. The folder 'PPC/code/fpga_parms' contains a parameter file for each of the nine FPGA's. When compiling for FPGA_A, the file 'PPC/code/fpga_params/fpga_a.h' should be modified for the appropriate parameters, and then it's contents should be placed in 'PPC/code/fpga_params.h'. The build utility automatically copies the correct fpga parameters to fpga_params.h for each FPGA that it compiles. The parameters found in fpga_params.h (and each file in the fpga_params folder) are as follows:

FPGA_NAME	Defines text used in 'print' statements to identify the FPGA
INTERCON_MASTER	If INTERCON_MASTER was defined in the synplicity project file, then it should be defined here to include the associated menu options. See the synplicity project file section above for more information.
INCLUDE_SRAM	Includes menu options associated with the SRAM device
INCLUDE_DDR1 EXCLUDE_DDR1	Includes menu options associated with DDR memory
INCLUDE_DDR2 EXCLUDE_DDR2	Expands the DDR test range to twice the size.
DDR_32_MEG DDR_64_MEG	Defines DDR test range per DDR chip (define one or the other, or none if neither DDR is included)

These files may be edited by hand, or modified with the build utility as follows:

make INCLUDE_DDR2	makes changes to synplicity projects, EDK source code, and UCF files to include DDR2
make EXCLUDE_DDR2	makes changes to synplicity projects, EDK source code, and UCF files to exclude DDR2
make DDR_32_MEG	makes changes to synplicity projects and EDK source code to set DDR size
make DDR_64_MEG	makes changes to synplicity projects and EDK source code to set DDR size
make DDR_128_MEG	makes changes to synplicity projects and EDK source code to set DDR size

V. Setting up the build utility: "make.bat"

The following tools must be installed on the system to use "make.bat":

- Xilinx ISE

- Xilinx EDK
- Synplicity Pro

In the section below %XILINX% should be replaced with your Xilinx install directory. By default this is "C:\Xilinx". %XILINX_EDK% should be replaced with your Xilinx EDK install directory. This is commonly "C:\Xilinx\EDK". %SYNPLICITY% should be replaced with your synplicity install directory. This is usually of the form "C:\Program Files\Synplicity\synplify_XX" where XX is the version number, like synplify_76 for synplify version 7.6.

The following directories must be in your "Path" environment variable:

- %XILINX%\bin\nt
- %XILINX_EDK%\gnu\powerpc-eabi\nt\bin;
- %XILINX_EDK%\xygwin\bin;
- %SYNPLICITY%\bin

At the bottom of each .prj file in the "synthesis" directory, is a line with the format:

- set_option -include_path "..."

Add the path "%SYNPLICITY%/lib/xilinx/" to this line if it is not already there.

2 Getting More Information

2.1 Printed Documentation

The printed documentation, as mentioned previously, takes the form of a Virtex-II Pro datasheet and a DN6000K10PCI User Guide.

2.2 Electronic Documentation

Multiple documents and datasheets have been included on the CD.

2.3 Online Documentation

There is a public access site that can be found on the Dini Group web site at <http://www.dinigroup.com/>.

Programming/Configuring the Hardware

This chapter details the programming and configuration instructions for the DN6000K10PCI.

1 Programming the Configuration FPGA

Note: The Configuration FPGA/PROM only needs to be programmed when an update is required.

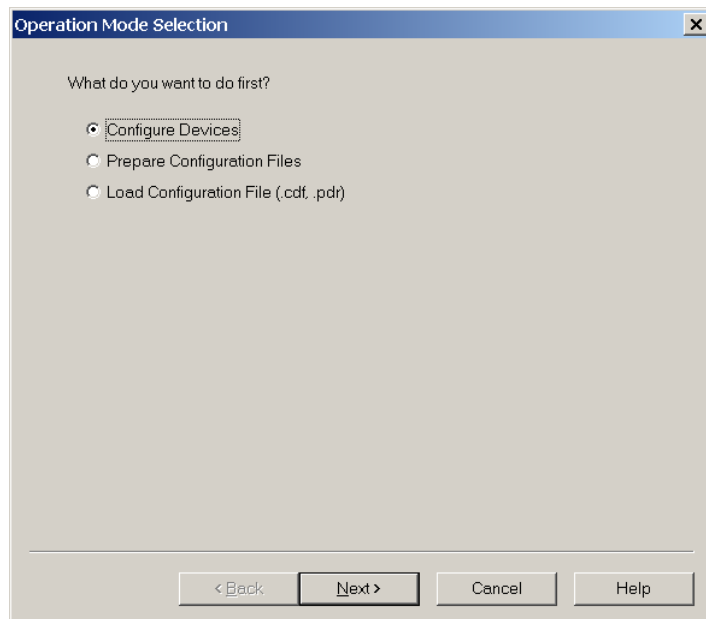
Code updates will be posted on the Dini Group website. The user is required to purchase the Xilinx Development Tools if in-house development is required. The tools are available from Xilinx, (<http://www.xilinx.com/>).

The Configuration FPGA (U2) is programmed using an in-system programmable configuration PROM (U66). The JTAG chain from the PROM is in a serial daisy chain with the Configuration FPGA, allowing simultaneous JTAG programming option of both devices. The Configuration FPGA is set to Master Serial Mode using dipswitch (S3). At power-up, the Configuration FPGA provides a configuration clock (CFPGA_CCLK) that drives the PROM. A short access time after CEN (CFPGA_DONE) and OE (CFPGA_INITn) are enabled, data is available on the PROM data (CFPGA_D0) pin that is connected to the Configuration FPGA. The programming header (J6) is used to download the files to the Configuration PROM/FPGA via a Xilinx Parallel IV cable.

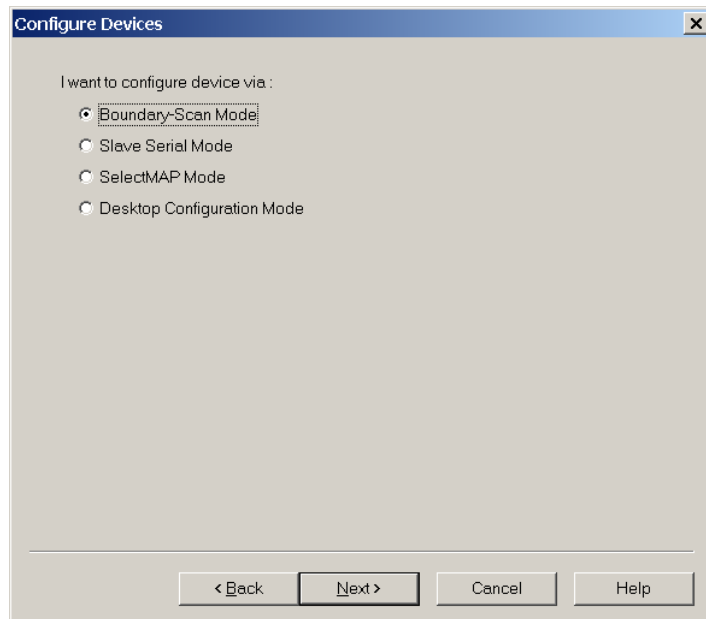
This section lists detailed instructions for programming the Configuration FPGA PROM using the Xilinx ISE 6.1i tools.

Note: This user guide will not be updated for every revision of the Xilinx tools, so please be aware of minor differences.

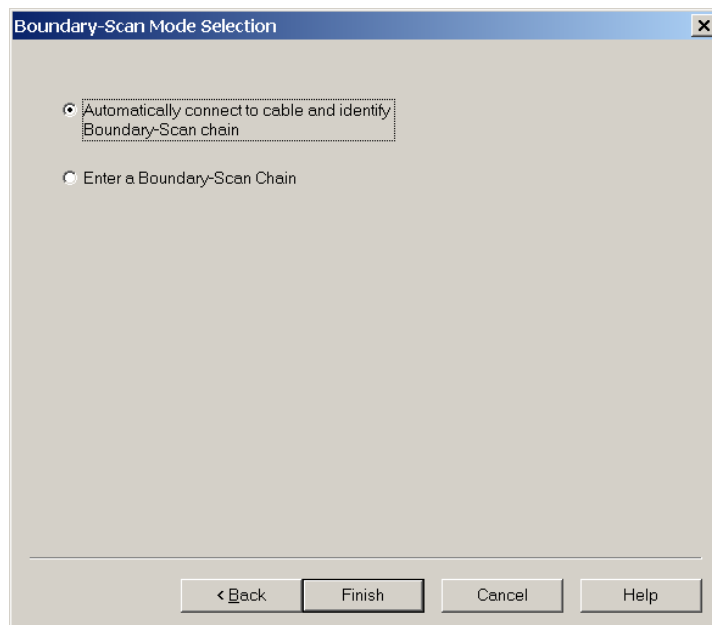
1. The DN6000K10PCI must be powered with the Xilinx JTAG cable connected to header J6 and the other end to a parallel port on the PC.
2. Download the latest programming file for the Configuration FPGA from the Dini Group website (filename "Prom.MCS") <http://www.dinigroup.com/>.
3. Run iMPACT - From the Windows **START** menu, choose **PROGRAMS → Xilinx ISE 6 → Accessories → iMPACT**.
4. Select the **Configure Devices** option and proceed by clicking the **NEXT** button.



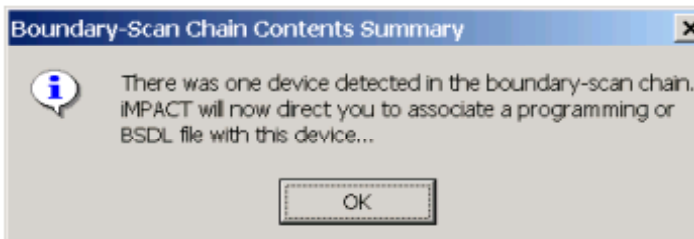
5. Select the **Boundary-Scan Mode** option and proceed by clicking the **NEXT** button.



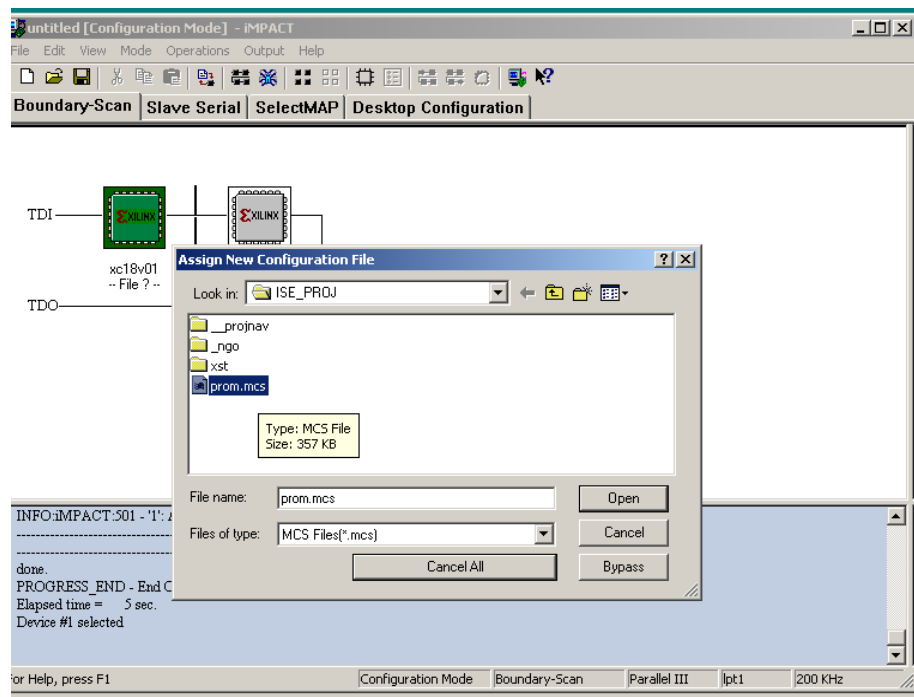
6. Select the **Automatically connect to cable and identify Boundary-Scan chain** option and proceed by clicking the **NEXT** button.



7. If the process was successful the following window will appear:

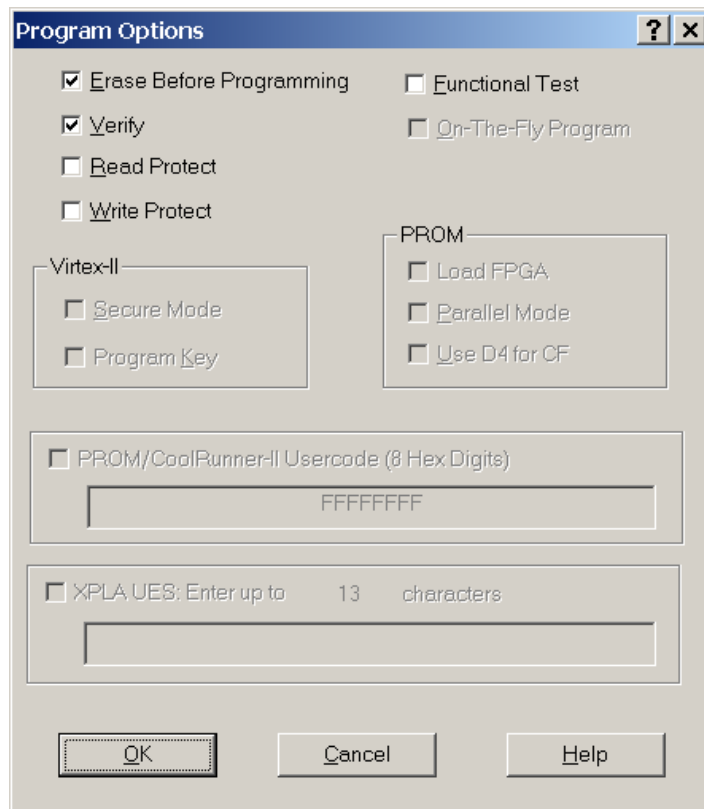


8. Click **OK** button.
9. Enter the location of the PROM.MCS file in the window prompting the file name and click **OK**. Select Bypass for the second device in the chain (XC2S150). The following window would be displayed:



Note: Two devices should be detected, XC18V01 and XC2S150.

10. Select the XC18V01 right click and select **Program** option. The XC2S150 is not programmed.
11. Select the **Erase before programming** and the **Erase** option before clicking the **OK** button.



12. The Configuration FPGA is now programmed. You must power cycle the board before the Configuration FPGA will be configured with the new PROM data.

2 MCU Details / Programming the MCU

Switch 4 on S2 tells MCU how to boot

- If the 4th switch position is ON then the MCU boot sequence will behave in the following manner:
 - (3) If the USB cable is plugged in when the DN6000k10 is powered-on/reset the MCU boots from the EEPROM (U8) and waits for USBController applicatin to send commands. In this case, the MCU FLASH firmware stored in U6 can be updated. In this state the MCU has limited USB functionality and cannot configure the FPGAs via USB/SmartMedia or perform many of the other USB GUI functions.
 - (4) If the USB cable is NOT plugged in when the DN6000k10 is powered-on/reset the MCU first boots from the EEPROM

(U8) and then automatically boots from the MCU FLASH (U6). In this case, the MCU FLASH can NOT be updated.

- If the 4th switch position is OFF then the MCU will always boot from the MCU FLASH (U6) regardless of whether the USB cable is plugged in or not. When the MCU has booted from the FLASH it has full USB and FPGA configuration functionality. This is the default factory setup as of 1/1/05. Please note you can NOT update the MCU FLASH in the switch position.

3 Configuring HyperTerminal

A terminal emulator is required to monitor MCU transactions and to interact with the embedded PowerPC processors in the Reference Design. The Dini Group suggests using the Windows-based program - HyperTerminal (Hypترم.exe). The configuration files for HyperTerminal “mcu_rs232.ht” and “ppc_rs232.ht” are supplied on the CD-ROM or can be downloaded from the Dini Group website.

The RS232 ports are configured with the following parameters:

- Bits per second: 19200
- Data bits: 8
- Parity: None
- Stop Bits: 1
- Flow control: None
- Terminal Emulation: VT100

Two cables converting the 5 x 2 header to a DB9 are shipped with the DN6000K10PCI. The 5 x 2 headers connect to the MCU RS232 header P7, and any of the four PPC RS232 headers P1, P3, P4, and P2. These headers are not keyed - ensure correct pin orientation as noted below.

Note: MCU RS232 Header P7 is not keyed. Ensure correct pin orientation. Pin 1 is indicated with a letter 1 on the board silkscreen, as well as a dot. Pin 1 on the 5 X 2 cable header is indicated with a triangular shape printed on the connector, and by a colored wire on the cable.

Two female-to-female RS232 cables are provided with the DN6000K10PCI. These cables will attach directly to the RS232 ports of a PC. The Dini Group suggests Jameco

as a possible supplier, (<http://www.jameco.com>). The part number is 132345. Male-to-female extension cables are part number 25700.

4 Configuring the FPGA using SelectMAP

The simplest mode of configuration for the DN6000K10PCI Virtex-II PRO FPGA involves the SelectMAP configuration method using a SmartMedia card. The DN6000K10PCI ships with two 32 MB SmartMedia cards. One of these SmartMedia cards contains reference design bit files produced for SelectMAP configuration, and a file named “main.txt” that sets the configuration options (see “[Creating Configuration File main.txt](#)”). The other SmartMedia card is empty and available for user applications. To configure the FPGA’s with the reference design, please skip to “[Starting SelectMAP Configuration](#)”.

Status messages are reported by the MCU via the RS232 serial port during FPGA configuration. It is **NOT** necessary to have the serial port connection in order to configure the FPGA’s in SelectMAP mode. However, if an error occurs during the configuration, the user would be able to identify possible problems by viewing the configuration status messages. See [Switch 4](#) on S2 tells MCU how to boot

- If the 4th switch position is ON then the MCU boot sequence will behave in the following manner:
 - (5) If the USB cable is plugged in when the DN6000k10 is powered-on/reset the MCU boots from the EEPROM (U8) and waits for USBController applicatin to send commands. In this case, the MCU FLASH firmware stored in U6 can be updated. In this state the MCU has limited USB functionality and cannot configure the FPGAs via USB/SmartMedia or perform many of the other USB GUI functions.
 - (6) If the USB cable is NOT plugged in when the DN6000k10 is powered-on/reset the MCU first boots from the EEPROM (U8) and then automatically boots from the MCU FLASH (U6). In this case, the MCU FLASH can NOT be updated.
- If the 4th switch position is OFF then the MCU will always boot from the MCU FLASH (U6) regardless of whether the USB cable is plugged in or not. When the MCU has booted from the FLASH it has full USB and FPGA configuration functionality. This is the default factory setup as of 1/1/05. Please note you can NOT update the MCU FLASH in the switch position.

Configuring HyperTerminal on how to setup the serial port.

4.1 Bit File Generation for SelectMAP Configuration

Configuring the DN6000K10PCI Virtex-II PRO FPGA requires the generation of bit files by the Xilinx ISE tools.

NOTE: This user guide will not be updated for every revision of the Xilinx tools, so please be aware of minor differences. The Xilinx ISE 6.1i revision is used here.

First, a project must be created. Open the Xilinx ISE **Project Navigator** software package. Go to the File menu and select **New Project**. A “New Project” dialog box will pop up shown in [Figure 8](#).

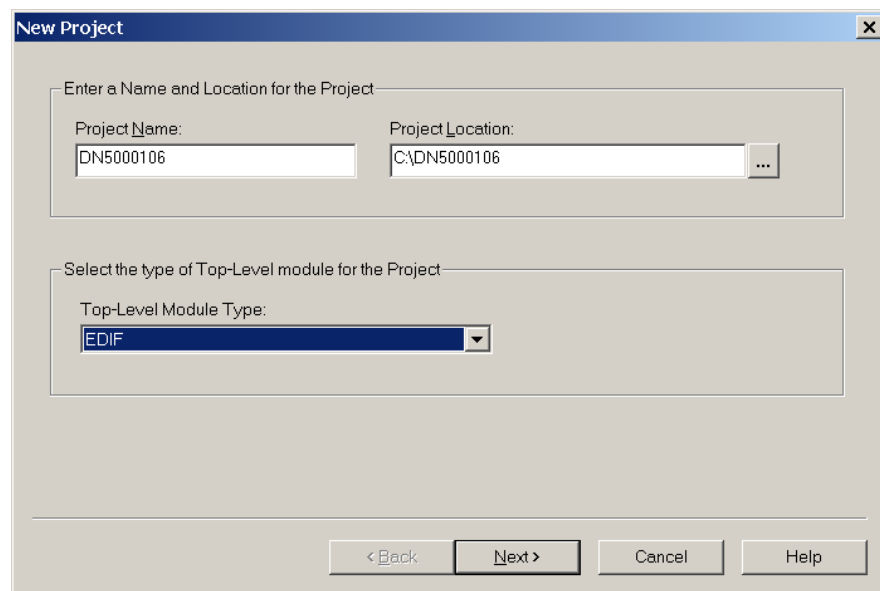


Figure 6 - New Project Screen Shot

Select the input files for the project, refer to [Figure 7](#).

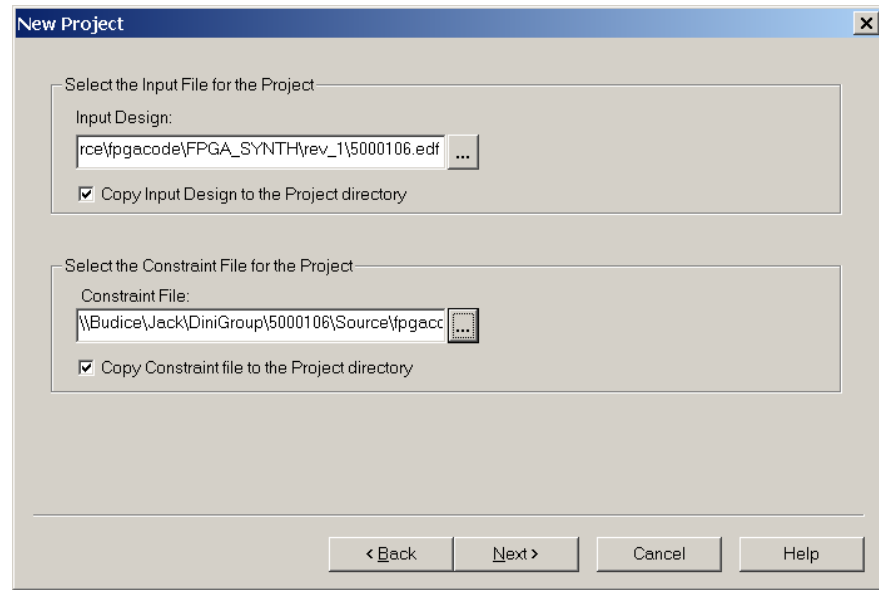


Figure 7 - Input File

Select the device and the design flow for the project. The user must specify a project name and location. The correct property values must be selected, refer to [Figure 8](#):

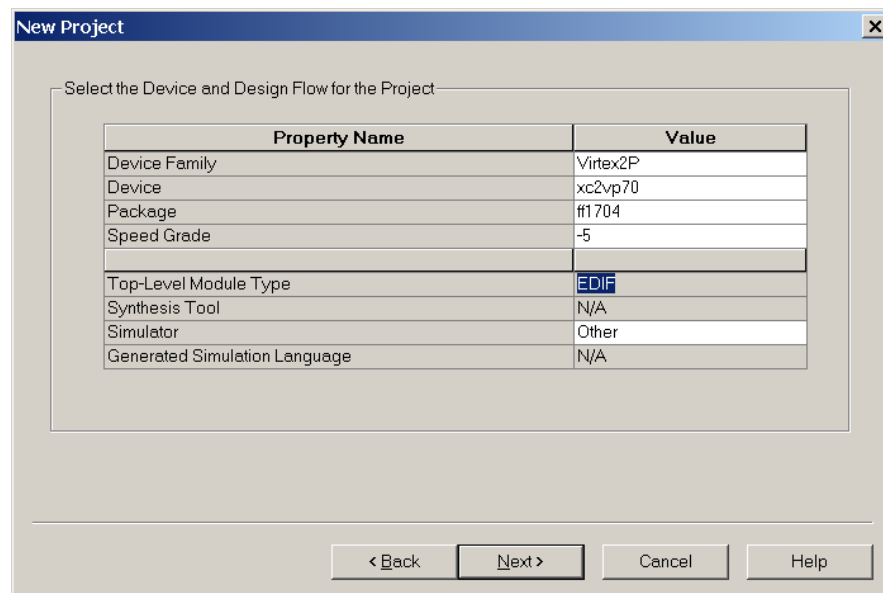


Figure 8: New Project Dialog Box

The Project Navigator will create a new project with the required files.

The DINI Group prefers to use Synplicity's Synplify for synthesis (which is recommended for the user also). Consequently, edif files are used in the design flow described here.

Selecting the edif file in the “Module View” window, the user’s Project Navigator box should resemble [Figure 9](#).

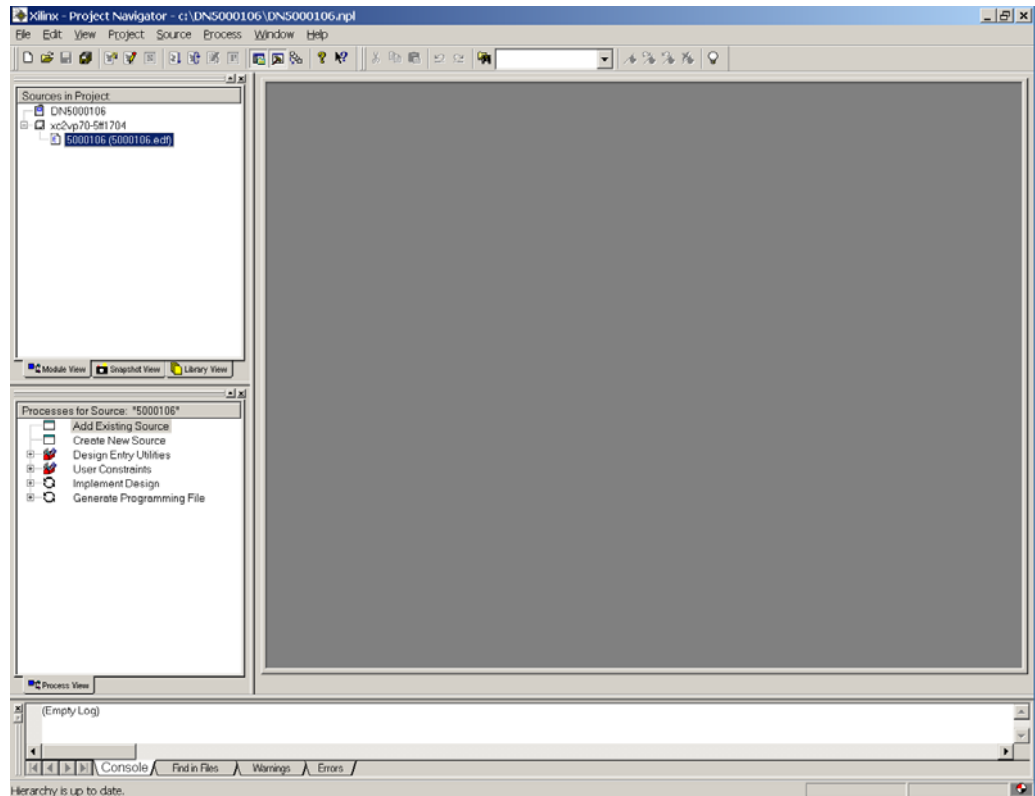
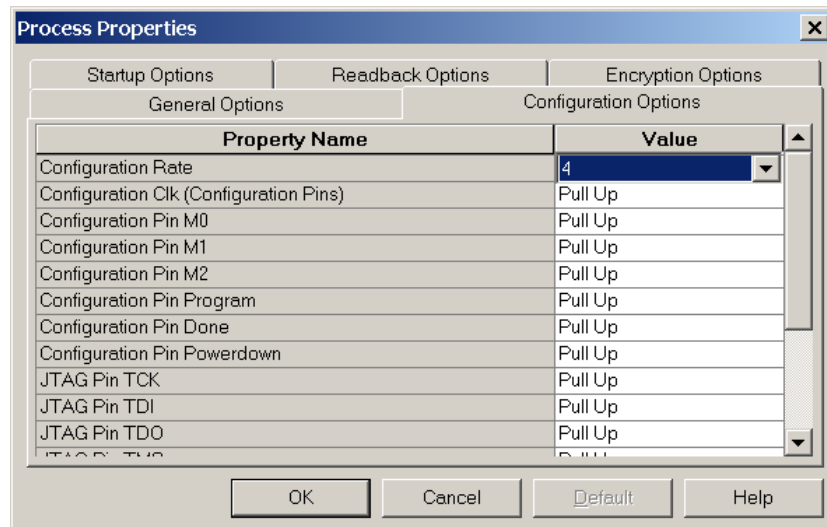


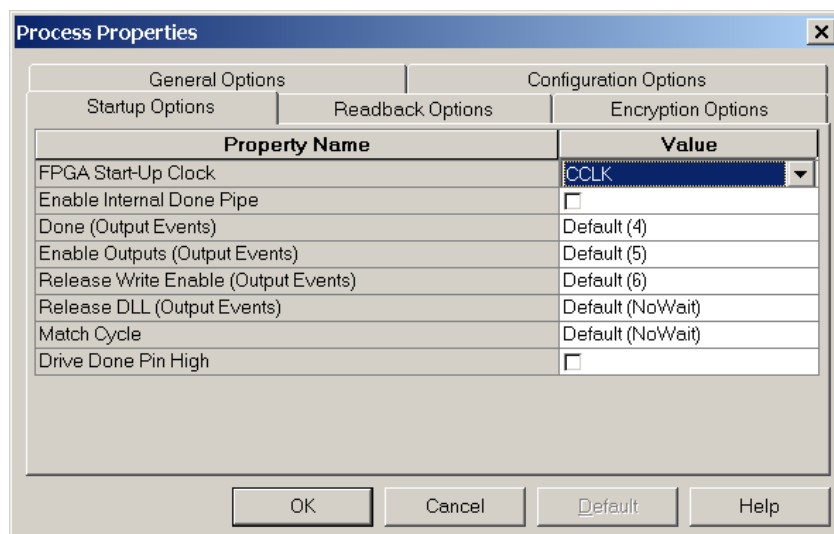
Figure 9: Project Navigator

In the “Process for Source” window, a process is signified by the icon . In the “Process for Source” window, the user must right-click on the “Generate Programming File” process and select properties. The default settings are correct (The user should verify a couple important options, right-click and selecting properties options).

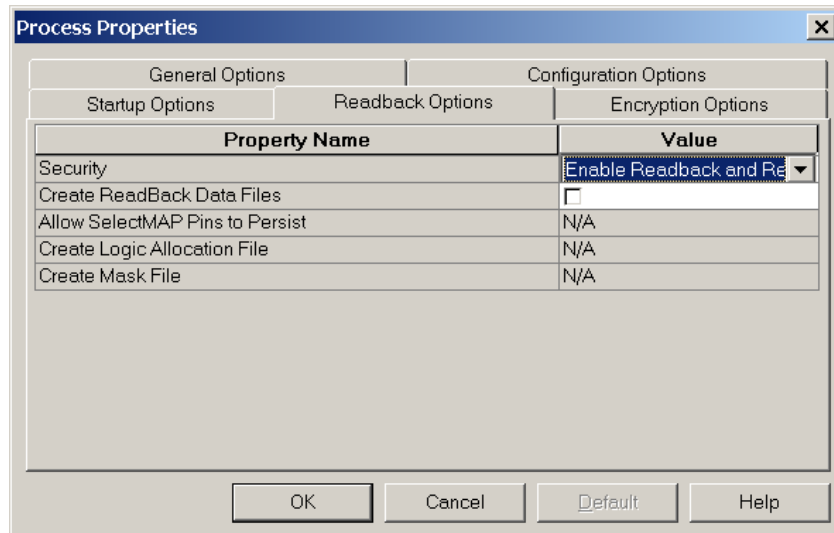
- Configuration Options Tab: Configuration Pin Powerdown = **Pull Up**



- Startup Options Tab: FPGA Start-up Clock = **CCLK**



- Readback Options Tab: Security = **Enable Readback and Reconfiguration**



The user can now generate the bit file. In the “Process for Source” window, the user must right-click on the “Generate Programming File” process and select Run. The bit file will be generated and may be found in the project directory.

4.2 Creating Configuration File “main.txt”

To control which bit file on the Smart Media card is used to configure which FPGA in SelectMAP mode a file named “main.txt” must be created and copied to the root directory of the Smart Media card. The configuration process cannot be performed without this file. Below is a description of the options that can be set in the file, a description of the format this file needs to follow, and an example of a main.txt file.

4.2.1 Verbose Level

During the configuration process, there are three different verbose levels that can be selected for the serial port messages:

- Level 0:
 - Fatal error messages
 - Bit file errors (e.g., bit file was created for the wrong part, bit file was created with wrong version of Xilinx tools, or bitgen options are set incorrectly)
 - Initializing message will appear before configuration
 - A single message will appear once the FPGA is configured
- Level 1:
 - All messages that Level 0 displays
 - Displays configuration type (should be SelectMAP)

- Displays current FPGA being configured if the configuration type is set to SelectMAP
- Displays a message at the completion of configuration for each FPGA configured.
- Level 2:
 - All messages that Level 1 displays
 - Options that are found in “main.txt”
 - Bit file names for each FPGA as entered in main.txt
 - Maker ID, device ID, and size of Smart Media card
 - All files found on Smart Media card
 - If sanity check is chosen, the bit file attributes will be displayed (part, package, date, and time of the bit file)
 - During configuration, a “.” will be printed out after each block (16 KB) has successfully been transferred from the Smart Media to the current FPGA

4.2.2 Sanity Check

The Sanity Check, if enabled, verifies that the bit file was created for the right part, the right version of Xilinx was used, and the bitgen options were set correctly. If any of the settings found in the bit file are not compatible with the FPGA, a message will appear from the serial port, and the user will be asked whether or not they want to continue with the bit file. Please see the section [Bit File Generation for SelectMAP Configuration](#) for details on which bitgen options need to be changed from the default settings..

4.2.3 Format of “main.txt”

The format of the main.txt file is as follows:

1. The first nonempty/uncommented line in main.txt should be:

Verbose level: X

where “X” can be 0, 1 or 2. If this line is missing or X is an invalid level, then the default verbose level will be 2.

2. The second nonempty/uncommented line in main.txt tells whether or not to perform a sanity check on the bit files before configuring an FPGA:

Sanity check: y

where “y” stands for yes, “n” for no. If the line is missing or the character after the “:” is not “y” or “n” then the sanity check will be enabled.

3. For each FPGA that the user wants to configure, there should be exactly one entry in the main.txt file with the following format:

FPGA F: example.bit

In the above format, the “F” following FPGA is to signal that this entry is for FPGA F, and FPGA F would then be configured with the bit file example.bit. The DN6000K10PCI only has one FPGA, which is FPGA F. There can be any number of spaces between the “:” and the configuration file name, but they need to be on the same line.

4. Comments are allowed with the following rules:

- All comments must start at the beginning of the line.
- All comments must begin with //
- If a comment spans multiple lines, then each line should start with //

Commented lines will be ignored during configuration, and are only for the user’s purpose.

5. The file main.txt is NOT case sensitive.
6. Example of “main.txt”:

```
//start of file “main.txt”
Verbose level: 2
Sanity check: y
FPGA F: fpgaF.bit
//the line above configures FPGA F with the bit file “fpgaF.bit”
//end of main.txt
```

Given the above example file: Verbose level is set to 2, a sanity check on the bit files will be performed, and FPGA F will be configured with file fpgaF.bit.

NOTE: All configuration file names have a maximum length of eight (8) characters, with an additional three for the extension. Do not name your configuration bit files with long file names. In addition, all file names should be located in the root directory of the Smart Media card—no subdirectories or folders are allowed. Since the “main.txt” file controls which bit file is used to configure the FPGA, the Smart Media card can contain other bit files.

4.3 Starting SelectMAP Configuration

If using the reference design SmartMedia card that came with the DN6000K10PCI then no files need to be copied to the card. Otherwise, copy your bit files and “main.txt” to the root directory of the SmartMedia card using the FlashPath floppy adapter or some other means. Make sure the dipswitch (S1) is set for SelectMAP as shown in [Table 2](#).

Table 2: S1 Dipswitch Configuration Settings

Signal Name	Pins	Status
FPGA_MSEL0	Pins 1 & 8	Closed
FPGA_MSEL1	Pins 2 & 7	Open
FPGA_MSEL2	Pins 3 & 6	Open
DIP_SW3	Pins 4 & 5	X

Set up the serial port connection as described above in [Switch 4](#) on S2 tells MCU how to boot

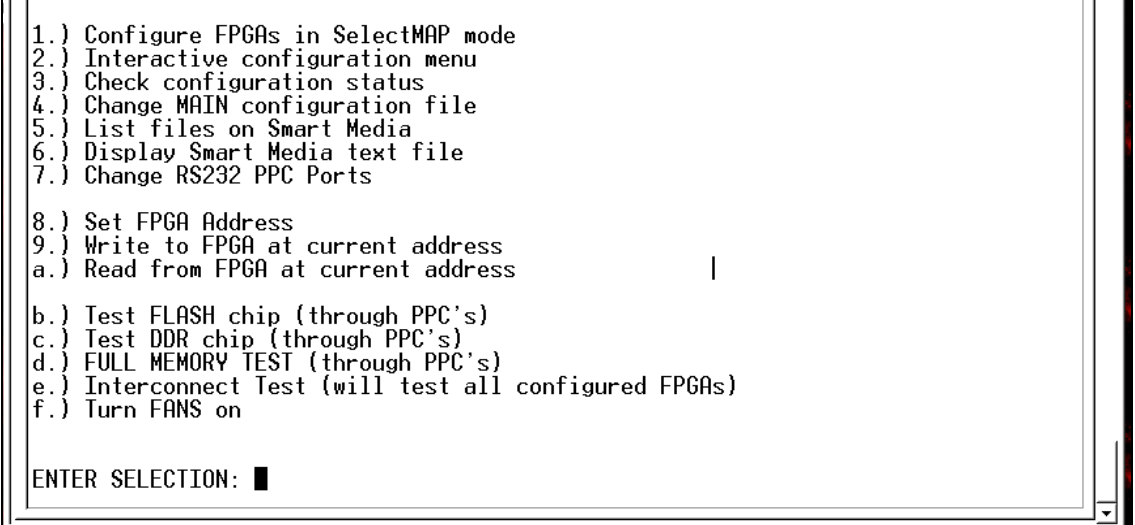
- If the 4th switch position is ON then the MCU boot sequence will behave in the following manner:
 - (7) If the USB cable is plugged in when the DN6000k10 is powered-on/reset the MCU boots from the EEPROM (U8) and waits for USBController applicatin to send commands. In this case, the MCU FLASH firmware stored in U6 can be updated. In this state the MCU has limited USB functionality and cannot configure the FPGAs via USB/SmartMedia or perform many of the other USB GUI functions.
 - (8) If the USB cable is NOT plugged in when the DN6000k10 is powered-on/reset the MCU first boots from the EEPROM (U8) and then automatically boots from the MCU FLASH (U6). In this case, the MCU FLASH can NOT be updated.
- If the 4th switch position is OFF then the MCU will always boot from the MCU FLASH (U6) regardless of whether the USB cable is plugged in or not. When the MCU has booted from the FLASH it has full USB and FPGA configuration functionality. This is the default factory setup as of 1/1/05. Please note you can NOT update the MCU FLASH in the switch position.

Configuring HyperTerminal. Next, place the SmartMedia card in the SmartMedia socket on the DN6000K10PCI and turn on the power (NOTE: the card can only go in one way). The SmartMedia card is hot swappable and can be taken out or put into the socket even when the power is on. Once the power has been turned on, the

configuration process will begin as long as there is a valid SmartMedia card inserted properly in the socket.

A SmartMedia card is determined to be invalid if either the format of the card does not follow the SSFDC specifications, or if it does not contain a file named `main.txt` in the root directory. If the configuration was successful, a message stating so will appear and the Main Menu will come up. Otherwise, an error message will appear. The LED's on DS1 and DS2 give feedback during and after the configuration process (see [Table 23](#) for GPIO LED's for further details).

After the FPGA has been configured, the following Main Menu will appear via the serial port, refer to [Figure 10](#).



```

1.) Configure FPGAs in SelectMAP mode
2.) Interactive configuration menu
3.) Check configuration status
4.) Change MAIN configuration file
5.) List files on Smart Media
6.) Display Smart Media text file
7.) Change RS232 PPC Ports

8.) Set FPGA Address
9.) Write to FPGA at current address
a.) Read from FPGA at current address

b.) Test FLASH chip (through PPC's)
c.) Test DDR chip (through PPC's)
d.) FULL MEMORY TEST (through PPC's)
e.) Interconnect Test (will test all configured FPGAs)
f.) Turn FANS on

ENTER SELECTION: 

```

Figure 10 - Main Menu

The HyperTerminal interface gives the user an easy method for handling and monitoring the DN6000K10PCI FPGA configuration.

4.3.1 Description of Main Menu Options

[Table 3](#) describes the Main Menu options found on the MCU HyperTerminal interface.

Table 3: HyperTerminal Main Menu Options

Option	Function	Description
1	Configure FPGA's Using "main.txt"	The FPGA will configure in SelectMAP mode.

Option	Function	Description
2	Interactive FPGA configuration menu	This option takes you to a menu titled “Interactive Configuration Menu” and allows the FPGA’s to be configured through a set of menu options instead of using the main.txt file. The menu options are described below.
3	Check Configuration Status	This option checks the status of the DONE pin and prints out whether or not the FPGA’s have been configured along with the file name that was used for configuration.
4	Change MAIN configuration file	By default, the processor uses the file main.txt to get the name of the bit file to be used for configuration as well as options for the configuration process. However, a user can put several files that follow the format for main.txt on the SmartMedia card that contain different options for the configuration process. By selecting the main menu option 4, the user can select a file from a list of files that can be used in place of main.txt. If the power is turned off or the reset button (S2) is pressed, the configuration file is changed back to the default, main.txt.
5	List files on SmartMedia	This option prints out a list of all the files found on the SmartMedia card.
6	Display Contents of a TXT File	This option allows the user to list the contents of any text file on the Smart Media card.
7	Change RS232 PPC Ports	This options allows the user to select what FPGA PPCs should be connected to which PPC PORTS (P1, P3, P4, & P7). This option will also print out the current port settings allowing you to quit without changing them. Please note that any single FPGA can only be hooked up to one PPC port. If you select an FPGA to be hooked up to more than one PPC Port then the first port will be able to transmit and received and all other ports will only receive.
The next 7 options are only available if the FPGAs are configured with The Dini Group reference design. Please see Appendix A for FPGA Address Maps.		
8	Set FPGA Address	Set the fpga address for the next read/write to the fpga.
9	Write to FPGA at current address	Performs a DWORD write to the current fpga address. You will see the current address at the top of the Main Menu and also the write data after selection this option.
a	Read from FPGA at current address	Performs a DWORD read at the current FPGA address. You will see the current address and readback data at the top of the Main Menu.
b	Test SRAM Chip (through PPC’s)	Allows the user to select which FPGA/SRAM to test. The test is actually run by the PPC’s and all detailed test messages will appear on PPC PORT 1 (P1)
c	Test DDR Chip (through PPC’s)	Allows the user to select which FPGA/DDR(s) to test. The test is actually run by the PPC’s and all detailed test messages will appear on PPC PORT 1 (P1)

Option	Function	Description
d	FULL MEMORY TEST (through PPC's)	Runs the following tests on all configured FPGAs: DDR, SRAM, Internal Registers, and Interconnect. The FPGA tests are performed in parallel and the user needs to hook up the PPC Ports to see detailed test messages. Only 4 FPGAs can output test messages via the PPC Ports so they will need to be setup before hand.
e	Interconnect Test	Runs the interconnect test on all configured FPGAs
f	Turn fans on/off	Either turns the fans on/off depending on current setting

Selecting “Option 2” results in the following menu to be displayed refer to [Figure 11](#).

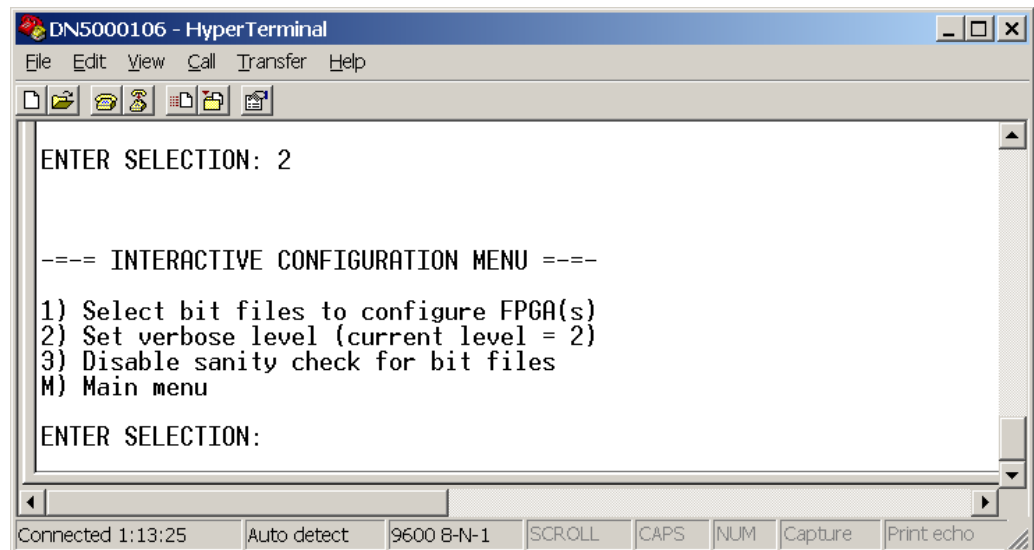


Figure 11 - Interactive Configuration Option Menu

[Table 4](#) describes the Interactive Configuration Menu options:

Table 4: HyperTerminal Interactive Configuration Menu Options

Option	Function	Description
1	Select a bit file to configure FPGA(s)	The user is able to select a bit file from a list of bit files found on the SmartMedia card for configuring the FPGA.

Option	Function	Description
2	Set verbose level (current level = 2)	The user can change the verbose level from the current setting. NOTE: If the user goes back to the main menu and configures the FPGA(s) using main.txt, the verbose level will be set to whatever setting is specified in main.txt.
3	Disable/Enable sanity check for bit files	The user can disable or enable the sanity check, depending on what the current setting is. NOTE: If the user goes back to the main menu and configures the FPGA(s) using main.txt, the sanity check will be set to whatever setting is specified in main.txt.
M	Main menu	Returns the user to the Main Menu.

4.4 Bitstream Encryption

Virtex-II Pro devices have an on-chip decryptor using one or two sets of three keys for triple-key Data Encryption Standard (DES) operation. Xilinx software tools offer an optional encryption of the configuration data (bitstream) with a triple- key DES determined by the designer. The keys are stored in the FPGA by JTAG instruction and retained by a battery connected to the VBAT_T pin, when the device is not powered. Virtex-II Pro devices can be configured with the corresponding encrypted bitstream, using any of the configuration modes described previously. A detailed description of how to use bitstream encryption is provided in the *Virtex-II Pro Platform FPGA User Guide*.

Board Hardware

1 Introduction to the Board

DN6000K10PCI Logic Emulation board provides for a comprehensive collection of peripherals to use in creating a system around the Virtex-II Pro FPGA. Figure 12 is a block diagram of the DN6000K10PCI Logic Emulation board diagram.

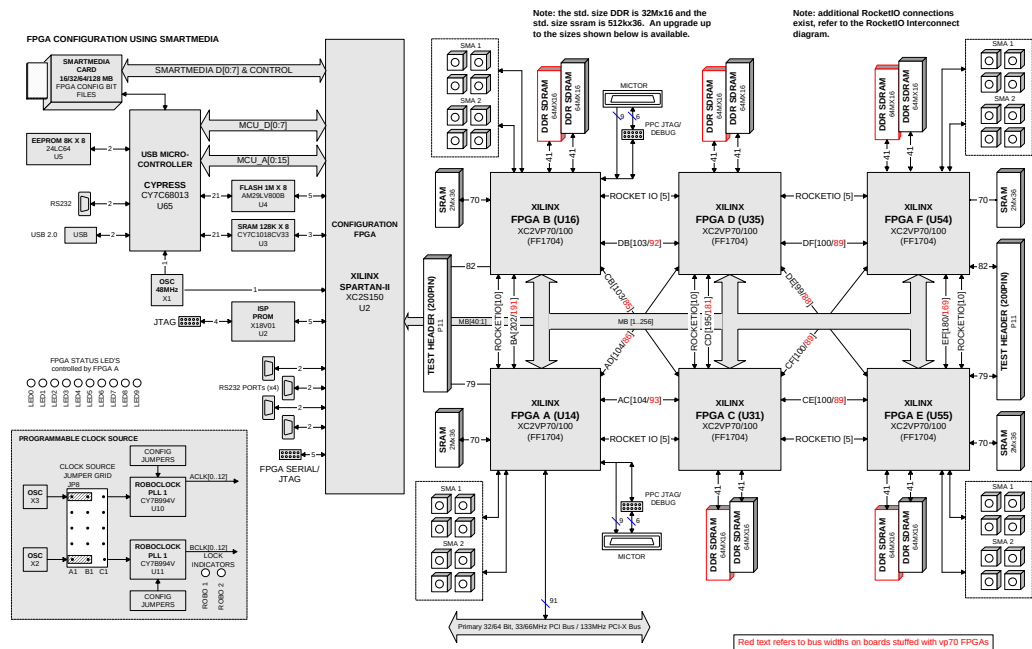


Figure 12 - DN6000K10PCI Block Diagram

1.1 DN6000K10PCI Functionality

The components and interfaces featured on the DN6000K10PCI include:

- 2VP70/100 Virtex-II Pro FPGA Options (x6)

- Flexible and Configurable Clocking Scheme (RoboClockII)
- SmartMedia FPGA Configuration
- USB2.0 Interface
- DDR SDRAM, 32M x 16 (size upgradeable to 64M x 16) -- Up to 2 on FPGA B,C,D,E,F
- SRAM, 512k x 36 (size upgradeable to 2M x 36) -- FPGA A,B,E,F
- Two Multi-Gigabit Transceiver (MGT) channels (SMB) / FPGA A, B, E, F
- One User Clock SMA Interface (differential SMB)
- 200 Pin Test Header (x 2)
- CPU Debug and Trace Interfaces, in Berg and Mictor connectors
- ATX Power Supply Connection

NOTE: RocketIO interface speed is directly affected by the speed grade of the FPGA. Please refer to the Xilinx datasheet.

2 Virtex-II Pro FPGA

The Virtex-II Pro FPGA's are situated on the topside of the board. For a detailed description of the capabilities of the Virtex-II Pro FPGA's, refer to the datasheet on the Xilinx website.

2.1 FPGA (2VP70) Facts

The Virtex-II Pro Platform FPGA's on board the DN6000K10PCI is in the FF1704 package. The capabilities of the 2VP70 (base model) include:

- 2 PowerPC™ 405 processor
- 16 or 20 Multi-Gigabit Transceivers (MGTs)
- 996 SelectI/O
- 8 Digital Clock Managers (DCMs)
- ~33000 logic slices

- ~5900 Kbits of BlockRAM (BRAM)
- 328 18 x 18-bit multiplier blocks

The FF1704 package on the DN6000K10PCI is a 1.0mm (42.5 x 42.5mm) fully populated (with four corner balls removed) flip chip BGA.

The PowerPCTM 405 is capable of operation at 300+ MHz, and is capable of 420+ Dhrystone MIPs (dependent on the speed grade of the part). Each of the MGTs are capable of 3.125 Gigabits per second in both directions, for an aggregate bandwidth of 50 Gigabits per second from the MGTs (25 Gbps transmit and 25 Gbps receive). The SelectIO are capable of supporting multiple high-speed I/O standards, from LVDS to SSTL2 to PCI. The DCMs are capable of 24 MHz to 420 MHz operation and provide for clock deskew, frequency synthesis, and fine phase shifting.

3 FPGA Configuration

The Dini Group developed the SmartMedia Configuration Environment to address the need for a space-efficient, pre-engineered, high-density configuration solution for systems with single or multiple FPGA's. The technology is a groundbreaking in-system programmable configuration solution that provides substantial savings in development effort and cost per bit over traditional PROM and embedded solutions for high-capacity FPGA systems.

Virtex-II Pro devices are configured by loading application-specific configuration data into internal memory. Configuration is carried out using a subset of the device pins, some of which are dedicated, while others can be reused as general-purpose inputs and outputs after configuration is complete. SmartMedia is the primary means of configuring the FPGA's on the DN6000K10PCI board. Configuration of FPGA's is accomplished using either Serial/SelectMAP or the JTAG interface. The remainder of this section describes the functional blocks that entail the FPGA configuration environment.

3.1 Micro Controller Unit (MCU)

The Cypress CY7C68013 (U65) micro controller is used to control the configuration process. The MCU contains an enhanced 8051 core, USB 2.0 transceiver and a Serial Interface Engine (SIE). The CY7C68013 provides the following features: 256 bytes of register RAM, three flexible Timers, 2 USARTs, and an integrated I²C compatible controller.

The MCU interfaces to the Configuration FPGA (U2) via an 8-bit bus and the SmartMedia interfaces to the Configuration FPGA via an 8-bit bus. The FPGA's (x6) on the board interfaces to the Configuration FPGA via the JTAG interface and an 8-bit bus, used during Serial and SelectMap programming of the FPGA's. The amount of

internal SRAM is not large enough to hold the FAT needed for SmartMedia, so an external 128Kb x 8 SRAM (U3) was added. In addition a 1Mb x 8 FLASH (U4) was added to store the downloaded program code. An external EEPROM (U5) configures the MCU during power-up.

The micro controller has the following responsibilities:

- Reading the SmartMedia card via the Configuration FPGA
- Communicate to the system via the USB Interface
- Configuring the Virtex-II Pro FPGA's (6)
- Executing DN6000K10PCI self tests
- Drive status LED's

3.1.1 MCU EEPROM Interface

During the power-up sequence, internal logic checks the I²C-compatible port for the connection of an EEPROM (U5) whose first byte is either 0xC0 or 0xC2. If found the MCU uses the VID/PID/DID values in the EEPROM in place of the internally stored values of it boot-loads the EEPROM contents into internal RAM (0xC2). The EEPROM interface is shown in Figure 13.

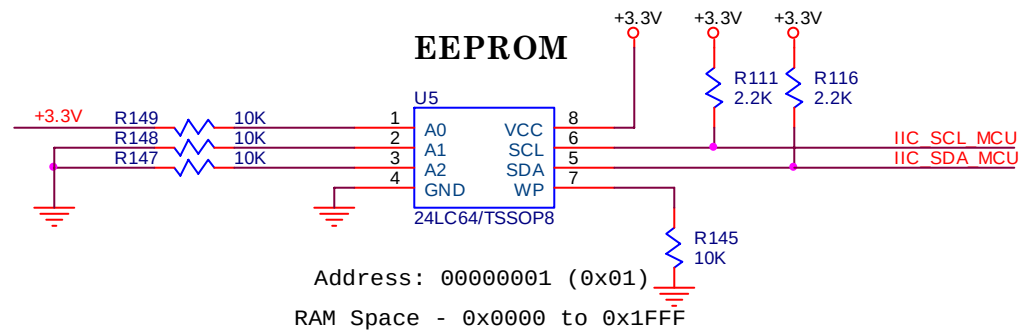


Figure 13 - MCU EEPROM Interface

3.1.2 MCU SRAM External

Memory expansion for the MCU is provided as 128k x 8 SRAM (U3). Writing to the device is accomplished by taking Chip Enable (SRAM_CS_n) and Write Enable (MEM_WR_n) inputs low. Reading from the device is accomplished by taking the Chip Enable (SRAM_CS_n) and the Output Enable (MEM_OE_n) low while forcing Write Enable high. The contents of the memory location specified by the address pins will appear on the IO pins. Address space above 2000H is banked through the Configuration FPGA. The SRAM interface is shown in Figure 14.

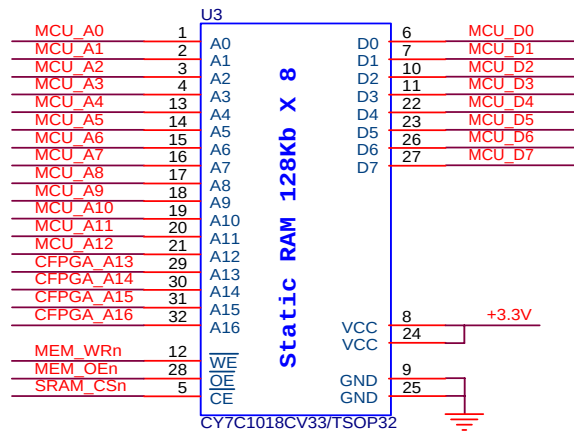


Figure 14 - MCU SRAM

3.1.3 MCU FLASH

Program memory is provided by the 1Mb x 8 FLASH (U4). To eliminate bus contention the device has separate Chip Enable (FLASH_CS_n), Write Enable (MEM_WR_n) and Output Enable (MEM_OE_n) controls. Device programming occurs by executing the program command sequence. Address space above 2000H is banked through the Configuration FPGA. The FLASH interface is shown in Figure 15.

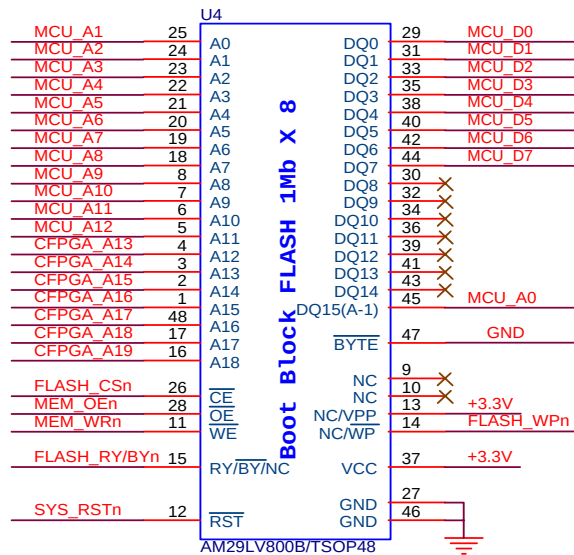


Figure 15 - MCU FLASH

3.1.4 MCU USB 2.0 Interface

Communication with the system is via the USB connector (J4), which interfaces directly with the MCU. The USB interface connector is a type B receptacle as shown in Figure 16.

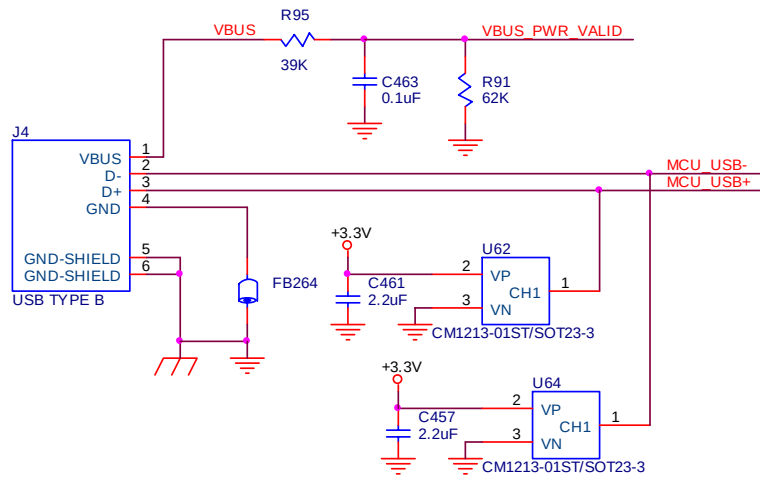


Figure 16 - USB Connector

3.1.5 RS232 Interface

An RS232 serial port (P7) is provided for low speed communication with the MCU. The RS-232 standard specifies output voltage levels between $-5V$ to $-15V$ for logical 1 and $+5V$ to $+15V$ for logical 0. Input must be compatible with voltages in the range of $-3V$ to $-15V$ for logical 1 and $+3V$ to $+15V$ for logical 0. This ensures data bits are read correctly even at maximum cable lengths between DTE and DCE, specified as 50 feet.

The RS-232 standard has two primary modes of operation, Data Terminal Equipment (DTE) and Data Communication Equipment (DCE). These can be thought of as host or PC for DTE and as peripheral for DCE. The DN6000K10PCI operates in the DCE mode only.

Figure 17 shows the implementation of the serial port on the DN6000K10

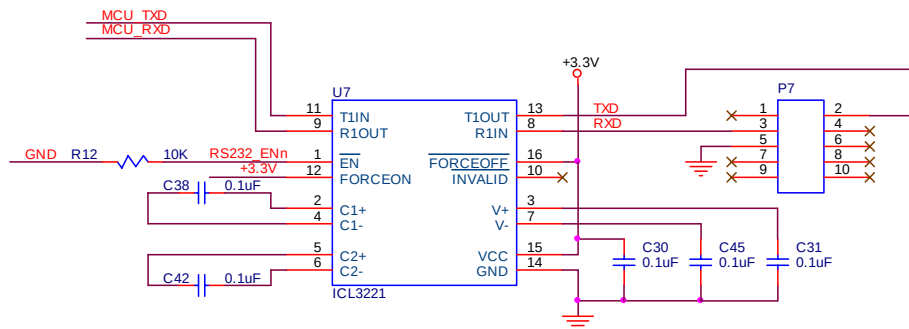


Figure 17 - MCU Serial Port

There are two signals attached to the MCU:

- Transmit Data

- Receive Data

TXD and RXD provide bi-directional transmission of transmit and receive data. No hardware handshaking is supported.

3.2 Configuration FPGA

The Xilinx Spartan-II XC2S150 (U2) is needed to handle the counters and state machines associated with the high-speed USB interface and the SmartMedia card. The FPGA contains 150K logic gates, 48K of BlockRAM and 260 user I/O's. The Verilog source code for the Configuration FPGA (ConfigFPGA.v) is provided on the CD-ROM.

The Configuration FPGA performs the following functions:

- Interface to the Micro Controller
 - Data Bus: MCU_D[0..7]
 - Address Signals: MCU_A[0..15]
 - Control Signals: MCU_RDn, MCU_WRn, MCU_CSn, MCU_OEn, MCU_PSEn
 - Clock: MCU_CLK
 - High Speed USB: SM_D[0..7], GPIF_RDYn, GPIF_CTL
- Interface to the SmartMedia
 - Data Bus: SM_D[0..7]
 - Control Signals: SM_REn, SM_WEn, SM_ALE, SM_CLE, SM_CEn, SM_RDYBUSYn
- Banked Address to the SRAM/FLASH
 - Upper Address Signals: CFPGA_A[13..19]
- FPGA Configuration, Serial/SelectMap
 - Data Bus for FPGA A,B,C: FPGA_1D[0..7]
 - Data Bus for FPGA D,E,F: FPGA_2D[0..7]
 - Data Bus for FPGA G,H,I: FPGA_3D[0..7]
 - Control Signals: FPGA_INIT_A, FPGA_DONE_A, FPGA_PROGn_A, FPGA_RD/WRn_A, FPGA_CSn_A, FPGA_BUSY_A, these signals are reproduced for FPGA A to FPGA I.
 - Clock: FPGA_DCLK
- FPGA Configuration, JTAG

- JTAG Signals: FPGA_TCK, FPGA_TDI, FPGA_DONE/TDO, FPGA_TMS_ABC, FPGA_TMS_DEF, FPGA_TMS_GHI
- SRAM Chip Select Generation
 - Signal: SRAM_CS_n
- FLASH Chip Select Generation
 - Signal: FLASH_CS_n
- FPGA Configuration MODE Select DipSwitch
 - Signals: FPGA_MSEL[0..3]
- Interface to the UART Connectors
 - RS232 Signals from the FPGA's: PPCA_TXD, PPCA_RXD,.....PPCI_TXD, PPCI_RXD, for FPGA A to I.
 - RS232 Signals to the Connectors: PPC_TXD1, PPC_TXD2, PPC_TXD3, PPC_TXD4, PPC_RXD1, PPC_RXD2, PPC_RXD3, PPC_RXD4, PPC_MON1, PPC_MON2.
- LED Indicators
 - Signals: CFPGA_LED_n[0..3]
- GPIO between Configuration FPGA and FPGA's (6)
 - Signals: MB[1..40]

3.2.1 Configuration PROM/FPGA Programming

The Configuration FPGA (U2) is programmed using an in-system programmable configuration PROM (U66). The JTAG chain from the PROM is in a serial daisy chain with the Configuration FPGA, allowing simultaneous JTAG programming option of both devices. The Configuration FPGA is set to Master Serial Mode using dipswitch (S3). At power-up, the Configuration FPGA provides a configuration clock (CFPGA_CCLK) that drives the PROM. A short access time after CEn (CFPGA_DONE) and OE (CFPGA_INIT_n) are enabled, data is available on the PROM data (CFPGA_D0) pin that is connected to the Configuration FPGA. The programming header (J6) as shown in [Figure 18](#), is used to download the files to the Configuration PROM/FPGA via a Xilinx Parallel IV cable.

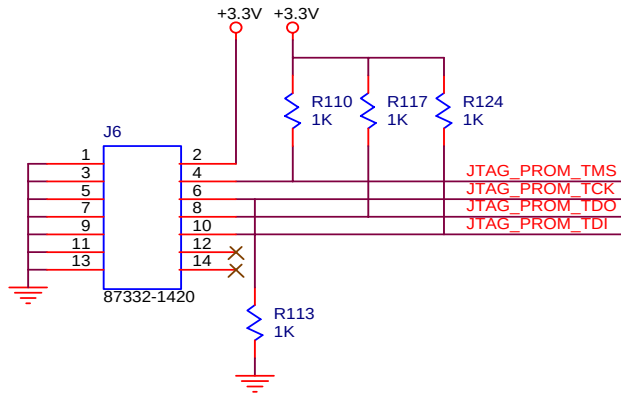


Figure 18 – Configuration PROM/FPGA Programming Header

3.2.2 Design Notes on the Configuration FPGA

Oscillator (X1) is a 48 MHz oscillator used to clock the Configuration FPGA. This part is soldered down to the PWB and is not intended to be user-configurable. The 48 MHz is divided down to 24 MHz in the Configuration FPGA to provide the clock for the micro controller (U65). The clock signal is labeled MCU_CLK on the schematic. The 48 MHz is used directly for the state machines in the Configuration FPGA for controlling the interface to the SmartMedia card. The maximum frequency for SelectMap configuration is 50 MHz without wait states.

Serial and JTAG configuration of the Virtex-II Pro FPGA's are back-off positions only. The 48 MHz clock can be divided down in the Configuration FPGA and used as a clock source to the PWB clock network (CFPGA_CLKOUT).

The signals MB[1..40] connects to the MB bus that links all the FPGA's. CFPGA_MSEL[0..2] selects the configuration mode of the Configuration FPGA (refer to Table 5) using dipswitch (S3).

Table 5 - FPGA Configuration Modes

Configuration Mode	M2	M1	M0	CLK Direction	Data Width	Serial Dout
Master Serial	0	0	0	Out	1	Yes
Slave Serial	1	1	1	In	1	Yes
Master SelectMAP	0	1	1	Out	8	No
Slave SelectMAP	1	1	0	In	8	No
Boundary Scan	1	0	1	N.A.	1	No

Note: Grayed options not supported by this design.

3.3 SmartMedia

The configuration bit file for the FPGA is copied to a SmartMedia card using the SmartDisk FlashPath Floppy Disk Adapter. The approximate file size for each possible FPGA option is shown below in [Table 6](#). Note that several BIT files can be put on a 32MB card. The DN6000K10PCI is shipped with two 32-megabyte 3.3V SmartMedia cards. The DN6000K10PCI supports card densities up to 128MB.

Note: Do **NOT** format the SmartMedia card using the default Windows file format program. Smart Media cards come pre-formatted from the factory, and files can be deleted from the card when they are no longer needed. If the SmartMedia card requires formatting, format the media with the program supplied by the FlashPath (SmartMedia floppy adapter) software.

Table 6 - FPGA configuration file sizes

Virtex-II Pro Device	Bitstream Length (bits)
XC2VP70	25,604,096
XC2VP100	33,645,312

SmartMedia Cards are available from www.computers4sure.com

3.3.1 SmartMedia Connector

[Figure 19](#) shows J1, the SmartMedia connector used to download the configuration files to the FPGA.

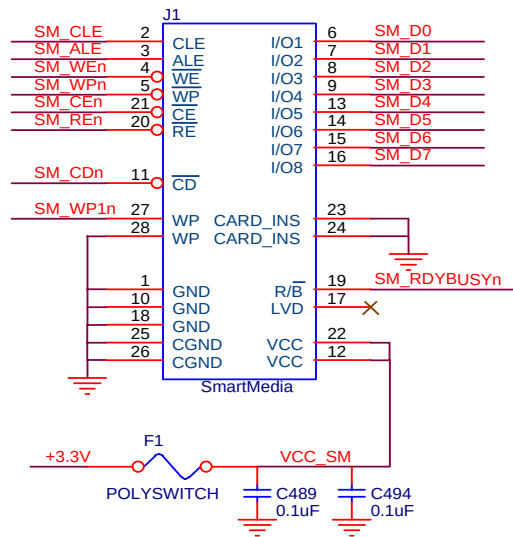


Figure 19 - SmartMedia Connector

Note: Do not press down on the top of the SmartMedia connector J1 if a SmartMedia card is not installed. The metal case shorts +3.3V to GND.

3.3.2 SmartMedia connection to Spartan (Configuration FPGA)/MCU

Table 7 shows the connection between the SmartMedia connector and the Configuration FPGA/MCU.

Table 7 - Connection between Configuration FPGA/MCU

Signal Name	Configuration FPGA/MCU	Connector
SM_D0	U2.K21	J1.6
SM_D1	U2.K22	J1.7
SM_D2	U2.J21	J1.8
SM_D3	U2.J20	J1.9
SM_D4	U2.J18	J1.13
SM_D5	U2.J22	J1.14
SM_D6	U2.J19	J1.15
SM_D7	U2.H19	J1.16
SM_CLE	U2.L20	J1.2
SM_ALE	U2.L17	J1.3

Signal Name	Configuration FPGA/MCU	Connector
SM_WEn	U2.L18	J1.4
SM_RDYBUSYn	U2.H18	J1.19
SM_CEn	U2.L21	J1.21
SM_REn	U2.L22	J1.20
SM_CDn	U65.106	J1.11
SM_WP1n	U65.82	J1.27

3.4 Boundary-Scan (JTAG, IEEE 1532) Mode

In boundary-scan mode, dedicated pins are used for configuring the Virtex-II Pro devices. The configuration is done entirely through the IEEE 1149.1 Test Access Port (TAP). The FPGA JTAG interfaces to IO on the Configuration FPGA. This allows manipulation of the data as required by the application and allows the JTAG chain to become an address on the existing bus. The processor can then read from, or write to the address representing the JTAG chain. FPGA's that are not populated requires feed through resistor to maintain the daisy chain connection between FPGA's.

3.4.1 FPGA JTAG Connector

Figure 20 shows J3, the JTAG connector used to download the configuration files to the FPGA's.

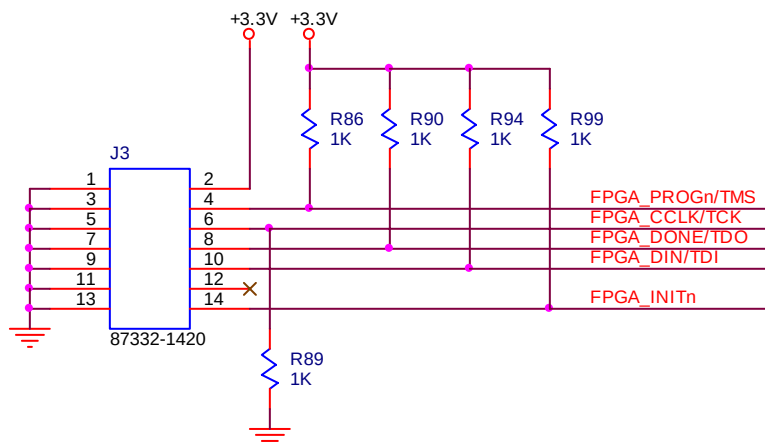


Figure 20 - FPGA JTAG Connector

3.4.2 FPGA JTAG connection to Configuration FPGA

Table 8 shows the connection between the FPGA JTAG connector and the Configuration FPGA.

on FPGA

Configuration FPGA	Connector
U2.N21	J3.6
U2.M20	J3.4
U2.M19	J3.8
U2.M18	J3.10
U2.M22	J3.14

ion

ulation board has a flexible and configurable clocking
ram showing the clocking resources and connections.

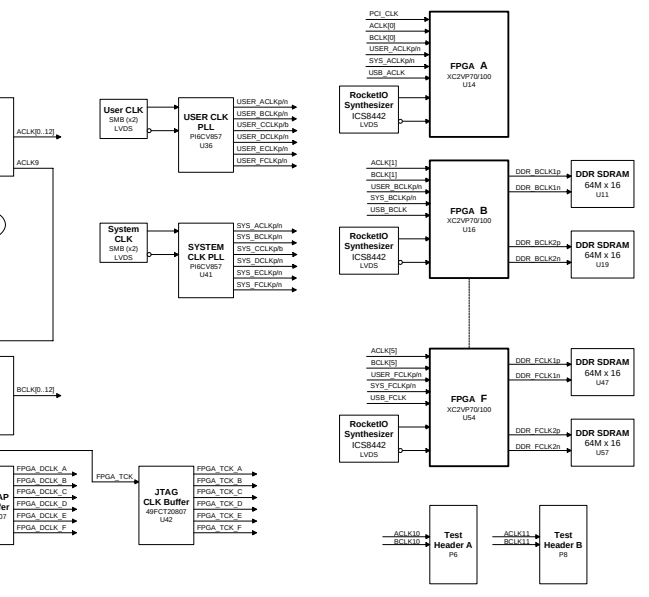


Figure 21 - Clocking Block Diagram

PN6000K10PCI include the following features:

- 7B994V) Multi-Phase PLL Clock Buffers

- External Differential User Clock Input (SMA Connectors J29/J4)
- System Oscillator (X4)
- Dedicated RocketIO Oscillators

The clock source selection grid formed by JP8, distributes clock signals (CLOCKA and CLOCKB) to two Roboclock PLL clock buffers (U10, U9). The clock outputs from the buffers are dispersed throughout the board. An external differential clock input option is available through the SMA connectors (J1, J4), which is the buffered and distributed throughout the board. A system oscillator (X4) is buffered and distributed throughout the board. This oscillator can be used to clock the Power PC's on each FPGA if required. Each FPGA has a dedicated RocketIO clock synthesizer driven by a 25MHz crystal. DDR clocks (DDR_CLKA....Ip/n) are generated by each individual FPGA. A dedicated 48MHz oscillator (X1) clocks the Configuration FPGA (U2), which in turn buffers the JTAG clock signal (FPGA_TCK) as well as the serial/parallel clock signal (FPGA_DCLK) required for FPGA configuration.

The connections between the FPGA's and various clocking resources are documented in [Table 9](#), covering the clocking inputs and outputs, respectively.

Table 9 - Clocking inputs to the FPGA's

Signal Name	FPGA A Pin	Clock Refdes and Pin
ACLK0	U14.K22	U10.89
BCLK0	U14.F22	U9.89
USER_ACLKp	U14.K21	U36.3
USER_ACLKn	U14.J21	U36.2
SYS_ACLKp	U14.AP21	U41.3
SYS_ACLKn	U14.AN21	U41.2
RCKTIO_OSCT_Ap	U14.F21	U20.14
RCKTIO_OSCT_An	U14.G21	U20.15
RCKTIO_OSCB_Ap	U14.AT21	U20.11
RCKTIO_OSCB_An	U14.AU21	U20.12
TST_HDRA_CLKIN	U14.AT22	P6.102
Signal Name	FPGA B Pin	Clock Refdes and Pin
ACLK1	U16.J22	U10.91
BCLK1	U16.G22	U9.91

USER_BCLKp	U16.AP21	U36.5
USER_BCLKn	U16.AN21	U36.6
SYS_BCLKp	U16.K21	U41.5
SYS_BCLKn	U16.J21	U41.6
DDR_BCLKp	U16.AU22	U15.5
DDR_BCLKn	U16.AT22	U15.6
RCKTIO_OSCT_Bp	U16.F21	U22.14
RCKTIO_OSCT_Bn	U16.G21	U22.15
RCKTIO_OSCB_Bp	U16.AT21	U22.11
RCKTIO_OSCB_Bn	U16.AU21	U22.12
Signal Name	FPGA C Pin	Clock Refdes and Pin
ACLK2	U31.AP21	U10.94
BCLK2	U31.AN21	U9.94
USER_CCLKp	U31.J22	U36.10
USER_CCLKn	U31.K22	U36.9
SYS_CCLKp	U31.AU22	U41.10
SYS_CCLKn	U31.AT22	U41.9
RCKTIO_OSCT_Cp	U31.G22	U45.14
RCKTIO_OSCT_Cn	U31.F22	U45.15
RCKTIO_OSCB_Cp	U31.AT21	U45.11
RCKTIO_OSCB_Cn	U31.AU21	U45.12
DDR_CCLKp	U31.K21	U32.5
DDR_CCLKn	U31.J21	U32.6
Signal Name	FPGA D Pin	Clock Refdes and Pin
ACLK3	U35.K21	U10.96
BCLK3	U35.F21	U9.96
USER_DCLKp	U35.AN22	U36.20
USER_DCLKn	U35.AP22	U36.29
SYS_DCLKp	U35.J22	U41.20
SYS_DCLKn	U35.K22	U41.19

RCKTIO_OSCT_Dp	U35.G22	U27.14
RCKTIO_OSCT_Dn	U35.F22	U27.15
RCKTIO_OSCB_Dp	U35.AU22	U27.11
RCKTIO_OSCB_Dn	U35.AT22	U27.12
DDR_DCLKp	U35.AT21	U34. 5
DDR_DCLKn	U35.AU21	U34.6
Signal Name	FPGA E Pin	Clock Refdes and Pin
ACLK4	U55.AU22	U10.66
BCLK4	U55.AN22	U9.66
USER_ECLKp	U55.K21	U36.22
USER_ECLKn	U55.J21	U36.23
SYS_ECLKp	U55.AP21	U41.22
SYS_ECLKn	U55.AN21	U41.23
DDR_ECLKp	U55.K21	U53.5
DDR_ECLKn	U55.J21	U53.6
RCKTIO_OSCT_Ep	U55.F21	U56.14
RCKTIO_OSCT_En	U55.G21	U56.15
RCKTIO_OSCB_Ep	U55.AT21	U56.11
RCKTIO_OSCB_En	U55.AU21	U56.12
Signal Name	FPGA F Pin	Clock Refdes and Pin
ACLK5	U54.K21	U10.64
BCLK5	U54.F21	U9.64
USER_FCLKp	U54.AN22	U36.46
USER_FCLKn	U54.AP22	U36.47
SYS_FCLKp	U54.J22	U41.46
SYS_FCLKn	U54.K22	U41.47
RCKTIO_OSCT_Fp	U54.G22	U59.14
RCKTIO_OSCT_Fn	U54.F22	U59.15
RCKTIO_OSCB_Fp	U54.AU22	U59.11
RCKTIO_OSCB_Fn	U54.AT22	U59.12

DDR_FCLKp	U54.AT21	U51.5
DDR_FCLKn	U54.AU21	U51.6

4.2 Clock Source Jumpers

The clock source grid JP8 gives the user the ability to select the clock input source to the RoboClock PLL buffers. A brief description of each pin is given in [Table 10](#).

Table 10 - Clock Source Signals

Signal Name	Description	Connector
CFPGA_CLKOUT	Clock signal from the Configuration FPGA.	JP8.A3
CLOCKA	Clock signal from oscillator X3	JP8.A1
CLOCKB	Clock signal from oscillator X2	JP8.A5
PLL1B	Secondary clock input to RoboClock, differential pair with PLL1BN	JP8.B4
PLL1BN	Secondary clock input to RoboClock, differential pair with PLL1B	JP8.B5
PLL2B	Secondary clock input to RoboClock, differential pair with PLL2BN	JP8.B1
PLL2BN	Secondary clock input to RoboClock, differential pair with PLL2B	JP8.B2
GND	Provides a ground reference for signals in the ribbon cable.	JP8.C1..C5

The PLL clock buffers can accept either LVTTTL33 or Differential (LVPECL) reference inputs (refer to [Figure 22](#)).

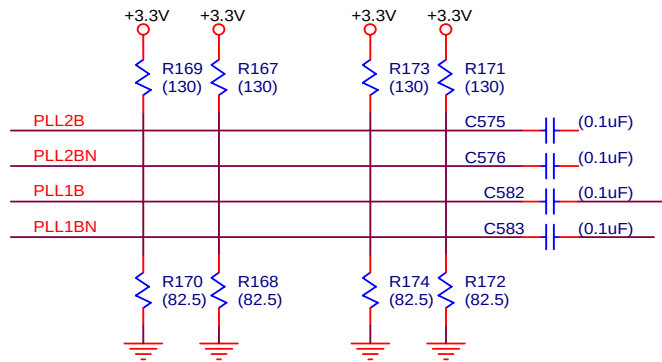


Figure 22 - LVPECL Clock Input and Termination

Note: The schematic shows capacitors in locations C582, C583, C575, and C576. These are actually populated with 0-ohm resistors for direct connection to the RoboClock reference inputs. The terminating resistors to GND and +3.3V are not stuffed. When using LVPECL, make the required hardware changes.

4.2.1 Clock Source Jumper Header

Figure 23 shows JP8, the clock source header connector used to select between different clock sources.

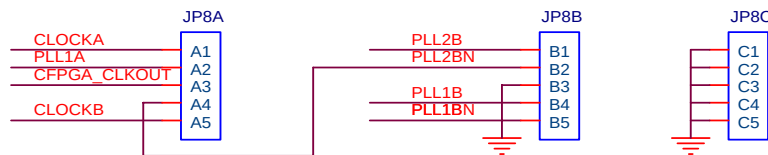


Figure 23 - Clock Source Jumper

4.3 Roboclocks

Two 3.3V half-can oscillator sockets (X2, X3) and the signal CFPGA_CLKOUT from the Configuration FPGA provide on-board input clock solutions. The DN6000K10PCI is shipped with both a 14.318MHz (X3) and a 33.33MHz (X2) oscillator. Neither X2 nor X3 are used by the configuration circuitry, so the user is free to stuff any standard 3.3 V half-can oscillators in the X2 and X3 positions. The oscillators interface to two high-speed multi-phase RoboClock buffers.

4.3.1 RoboClock PLL Clock Buffers

The CY7B994V (U10, U9) High-Speed Multi-Phase PLL Clock Buffers offer user-selectable control over system clock functions. Each chip has 16 output clocks along with two feedback output clocks. Two sets of eight output clocks are jumper selectable for each chip. The feedback clocks are controlled separately.

Eighteen configurable outputs each drive terminated transmission lines with impedances as low as 50 while delivering minimal and specified output skews at LVTTTL levels (refer to [Figure 24](#)). The outputs are arranged in five banks. Banks 1 to 4 of four outputs allow a divide function of 1 to 12, while simultaneously allowing phase adjustments in 625 ps - 1300 ps increments up to 10.4 ns. One of the output banks also includes an independent clock invert function. The feedback bank consists of two outputs, which allows divide-by functionality from 1 to 12 and limited phase adjustments. Any one of these eighteen outputs can be connected to the feedback input as well as driving other inputs.

Selectable reference input is a fault tolerance feature, which allows smooth change over to secondary clock source, when the primary clock source is not in operation. The reference inputs and feedback inputs are configurable to accommodate either LVTTTL or Differential (LVPECL) inputs. The completely integrated PLL reduces jitter. Please refer to the datasheet for more detailed information.

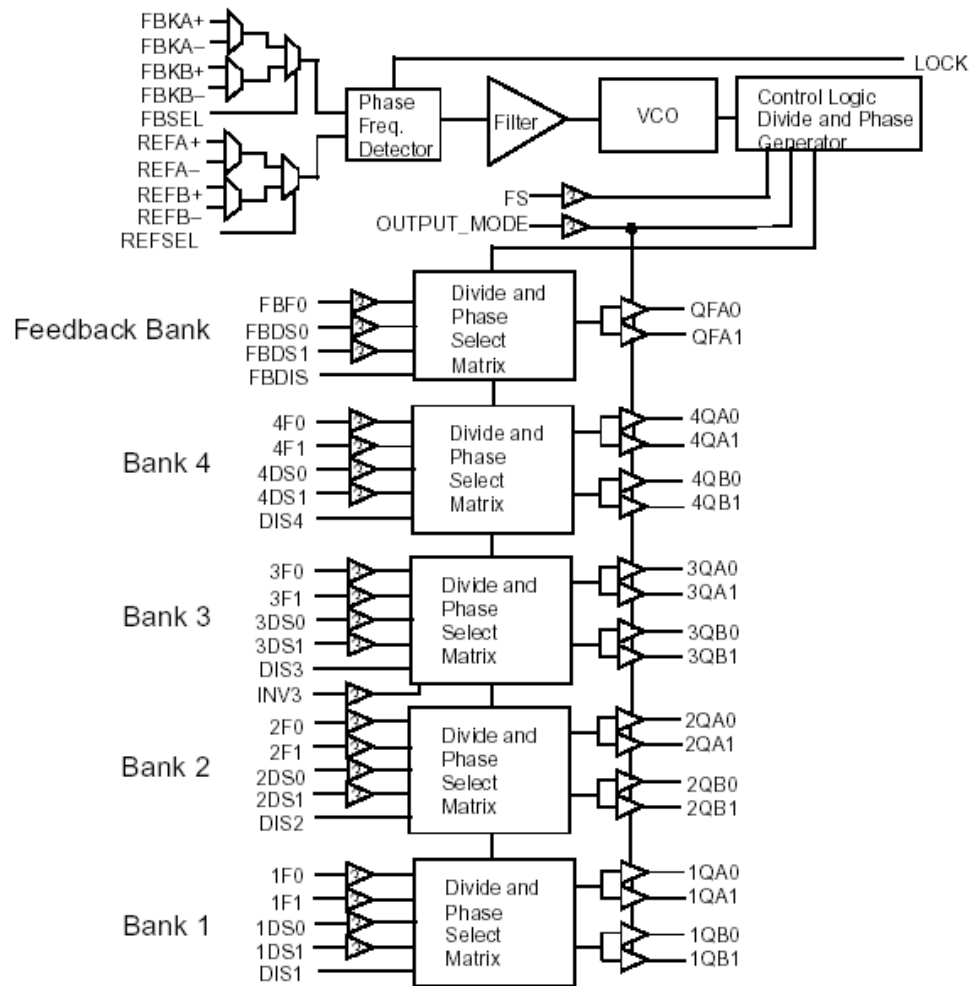


Figure 24 - RoboClock Functional Block Diagram

4.3.2 RoboClock Configuration Jumpers

Header JP6, JP4, and JP5 enable the user to configure the RoboClocks as required. These are 3-way headers and allow the signal to float (MID), or be pulled to GND (LOW) or +3.3V (HIGH). A brief description of each pin is given in [Table 11](#).

Table 11 - RoboClock Configuration Signals

Signal Name	Description	Connector
OSCA	Enable for Oscillator A (X9)	JP4.B1
OSCB	Enable for Oscillator B (X8)	JP4.B1
ROBO1_DIS	ROBOCLOCK #1, Output Disable: Each input controls the state of the respective output bank. When HIGH, the output bank is disabled to the “HOLD-OFF” or “HI-Z” state; the disable state is determined by	JP4.B5

Signal Name	Description	Connector
	OUTPUT_MODE. When LOW, the [1:4]Q[A:B][0:1] is enabled. (See <i>Table 5 in Datasheet</i>). These inputs each have an internal pull-down.	
ROBO2_DIS	ROBOCLOCK #2, Output Disable: Each input controls the state of the respective output bank. When HIGH, the output bank is disabled to the “HOLD-OFF” or “HI-Z” state; the disable state is determined by OUTPUT_MODE. When LOW, the [1:4]Q[A:B][0:1] is enabled. (See <i>Table 5 in Datasheet</i>). These inputs each have an internal pull-down.	JP4.B6
ROBO1_MODE	ROBOCLOCK #1, Output Mode: This pin determines the clock outputs’ disable state. When this input is HIGH, the clock outputs will disable to high-impedance (HI-Z). When this input is LOW, the clock outputs will disable to “HOLD-OFF” mode. When in MID, the device will enter factory test mode.	JP4.B7
ROBO2_MODE	ROBOCLOCK #2, Output Mode: This pin determines the clock outputs’ disable state. When this input is HIGH, the clock outputs will disable to high-impedance (HI-Z). When this input is LOW, the clock outputs will disable to “HOLD-OFF” mode. When in MID, the device will enter factory test mode.	JP4.B8
ROBO2_REFSEL1	ROBOCLOCK #2, Reference Select Input: The REFSEL input controls how the reference input is configured. When LOW, it will use the REFA pair (PLL1A) as the reference input. When HIGH, it will use the REFB pair (PLL1BC, PLL1BNC) as the reference input. This input has an internal pull-down.	JP5.B1
ROBO2_FS	ROBOCLOCK #2, Frequency Select: This input must be set according to the nominal frequency (fNOM). Refer to Table 1 in the datasheet.	JP5.B2
ROBO2_FBF0	ROBOCLOCK #2, Feedback Output Phase	JP5.B3

Signal Name	Description	Connector
	Function Select: Controls the phase function of bank 3 & 4 (CCLK) of outputs, refer to Table 3 in the datasheet.	
ROBO2_FBDS0	ROBOCLOCK #2, Feedback Divider Function Select: These inputs determine the function of the QFA0 and QFA1 outputs. Refer to Table 4 in the datasheet.	JP5.B4
ROBO2_FBDS1	ROBOCLOCK #12 Feedback Divider Function Select: These inputs determine the function of the QFA0 and QFA1 outputs. Refer to Table 4 in the datasheet.	JP5.B5
ROBO2_FBDIS	ROBOCLOCK #2, Feedback Disable: This input controls the state of QFA[0:1]. When HIGH, the QFA[0:1] is disabled to the “HOLD-OFF” or “HI-Z” state; the disable state is determined by OUTPUT_MODE. When LOW, the QFA[0:1] is enabled. Refer to Table 5 in the datasheet. This input has an internal pull-down.	JP5.B6
ROBO2_F0	ROBOCLOCK #2, Output Phase Function Select: Controls the phase function of bank 1, 2, 3 & 4 (ACLK) of outputs. Refer to Table 3 in the datasheet.	JP5.B7
ROBO2_F1	ROBOCLOCK #2, Output Phase Function Select: Controls the phase function of bank 1, 2, 3 & 4 (DCLK) of outputs. Refer to Table 3 in the datasheet.	JP5.B8
ROBO2_DS0	ROBOCLOCK #2, Output Divider Function Select: Controls the divider function of bank 1, 2, 3 & 4 (ACLK) of outputs. Refer to Table 4 in the datasheet.	JP5.B9
ROBO2_DS1	ROBOCLOCK #2, Output Divider Function Select: Controls the divider function of bank 1, 2, 3 & 4 (ACLK) of outputs. Refer to Table 4 in the datasheet.	JP5.B10
ROBO1_REFSEL1	ROBOCLOCK #1, Reference Select Input: The REFSEL input controls how the reference input is configured. When LOW, it will use the REFA pair (PLL1A) as the	JP6.B1

Signal Name	Description	Connector
	reference input. When HIGH, it will use the REFB pair (PLL1BC, PLL1BNC) as the reference input. This input has an internal pull-down.	
ROBO1_FS	ROBOCLOCK #1, Frequency Select: This input must be set according to the nominal frequency (fNOM). Refer to Table 1 in the datasheet.	JP6.B2
ROBO1_FBF0	ROBOCLOCK #1, Feedback Output Phase Function Select: Controls the phase function of bank 3 & 4 (CCLK) of outputs, refer to Table 3 in the datasheet.	JP6.B3
ROBO1_FBDS0	ROBOCLOCK #1, Feedback Divider Function Select: These inputs determine the function of the QFA0 and QFA1 outputs. Refer to Table 4 in the datasheet.	JP6.B4
ROBO1_FBDS1	ROBOCLOCK #1, Feedback Divider Function Select: These inputs determine the function of the QFA0 and QFA1 outputs. Refer to Table 4 in the datasheet.	JP6.B5
ROBO1_FBDIS	ROBOCLOCK #1, Feedback Disable: This input controls the state of QFA[0:1]. When HIGH, the QFA[0:1] is disabled to the “HOLD-OFF” or “HI-Z” state; the disable state is determined by OUTPUT_MODE. When LOW, the QFA[0:1] is enabled. Refer to Table 5 in the datasheet. This input has an internal pull-down.	JP6.B6
ROBO1_F0	ROBOCLOCK #1, Output Phase Function Select: Controls the phase function of bank 1, 2, 3 & 4 (ACLK) of outputs. Refer to Table 3 in the datasheet.	JP6.B7
ROBO1_F1	ROBOCLOCK #1, Output Phase Function Select: Controls the phase function of bank 1, 2, 3 & 4 (DCLK) of outputs. Refer to Table 3 in the datasheet.	JP6.B8
ROBO1_DS0	ROBOCLOCK #1, Output Divider Function Select: Controls the divider function of bank 1, 2, 3 & 4 (ACLK) of outputs. Refer to Table	JP6.B9

Signal Name	Description	Connector
	4 in the datasheet.	
ROBO1_DS1	ROBOCLOCK #1, Output Divider Function Select: Controls the divider function of bank 1, 2, 3 & 4 (ACLK) of outputs. Refer to Table 4 in the datasheet.	JP6.B10

4.3.3 Roboclock Configuration Headers

Figure 25 shows JP6, JP4, and JP5, the RoboClock configuration headers.

RoboClock Configuration Jumpers

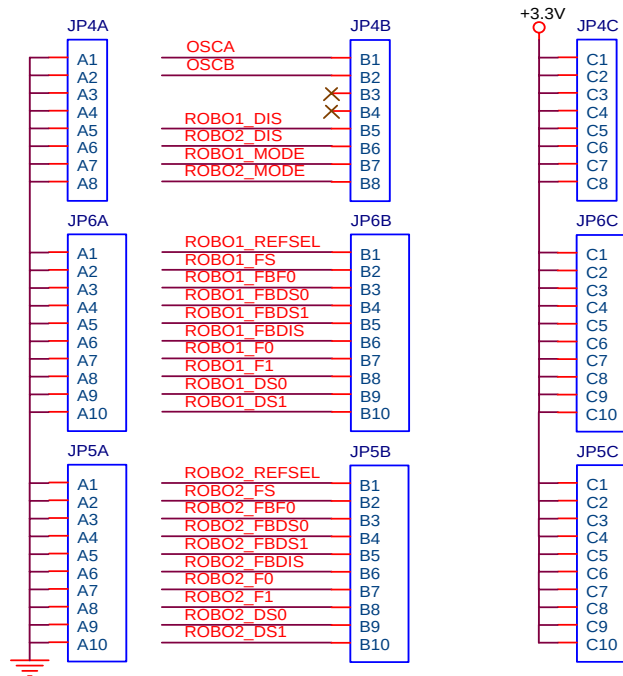


Figure 25 - RoboClock Configuration Jumpers

4.3.4 Useful Notes and Hints

The RoboClock consistently outputs ~32.5MHz signals in cases of improper settings or unacceptable clock inputs. This was observed when the CY7B994V part was operating at a nominal frequency f_{NOM} of 36.4MHz with FS set LOW. Identical clocks were sent to PLL2B and PLL2BN.

For the CY7B994V part, the operating frequency can reach up to 200 MHz. However, the maximum output frequency is 185MHz. This means when $185 \text{ MHz} < f_{\text{NOM}} < 200 \text{ MHz}$, the output divider must be set to at least 2. Otherwise, the RoboClocks will output garbage.

4.3.5 Customizing the Oscillators

The user can customize the frequency of the clock networks by stuffing different oscillators in X2 and X3. The DN6000K10PCI is shipped with a 14.318MHz oscillator in location X3 and a 33.333MHz oscillator in X2. The RoboClocks are not +5V tolerant, so +3.3V oscillators are necessary.

The Dini Group suggests Digi-Key (<http://www.digikey.com/>) as a possible source for the oscillators. Of note is the Epson line of oscillators called the SG-8002 Programmable Oscillators. Any frequency between 1.00MHz–106.25MHz can be procured in the normal Digi-Key shipping time of 24 hours. A half-can, +3.3 V CMOS version is needed with a tolerance of 50ppm. The part number for an acceptable oscillator from this family would be:

SG-8002DC-PCB-ND

- Package SG-8002DC (Halfcan)
- Output Enable
- 3.3 V CMOS
- $\pm\pm 50$ ppm

If the order is placed via the web page, the requested frequency to two decimal places is placed in the Web Order Notes. The datasheet is on the CD-ROM for this oscillator. Any polarity of output enabled for each oscillator (on pin 1) is acceptable. Ensure the proper jumper settings for JP6.B1/JP6.B2. See [Table 11](#) for a description.

4.3.6 Common Clock Source Selections

The following configuration is the most common:

Configuration 1: CLOCKS → PLL1A, CLOCKS → PLL2BN

RoboClock #1 (U62) is driven from oscillator X3. RoboClock #2 (U9) is driven from oscillator X2. RoboClock #2 can also be driven from RoboClock #1 output (ACLK9) if required.

4.4 External Clocks

The clock source jumper (JP8) allows the user a simple means to attach external clocks to the clock grid. The user can attach 10-pin ribbon cable to JP8B/C, which allows for connection the differential pair inputs of both RoboClocks. JP8C ground pins for signal integrity. These signals are described in [Table 10](#). Both differential pairs provide some flexibility. The user can bring a single 3.3V TTL input. It can be attached to either input. However, the other input must be left open. The user can provide a differential clock input to the pair to the RoboClocks.

4.4.1 External SMA Clock

J29/J30 are SMA connectors to allow an external differential clock (USER_CLKp/n) input to all the FPGA's via a PLL clock driver (U36). Resistors (R100, R116) allows for AC coupling if required. Refer [Figure 26](#).

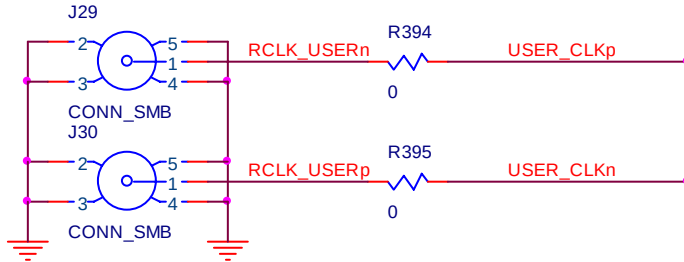


Figure 26 - External SMA Clock

4.4.2 Connections between FPGA's and External SMA Clock Buffer

The connection between the FPGA's and the external SMA clock buffer are shown in [Table 12](#).

Table 12 - Connection between FPGA and External PPC Oscillator

Signal Name	FPGA Pin	External SMA Clock Buffer
USER_ACLKp	U14.K21	U36.3
USER_ACLKn	U14.J21	U36.2
USER_BCLKp	U16.AP21	U36.5
USER_BCLKn	U16.AN21	U36.6
USER_CCLKp	U31.J22	U36.10
USER_CCLKn	U31.K22	U36.9
USER_DCLKp	U35.AN22	U36.20
USER_DCLKn	U35.AP22	U36.19
USER_ECLKp	U55.K21	U36.22
USER_ECLKn	U55.J21	U36.23
USER_FCLKp	U54.AN22	U36.46
USER_FCLKn	U54.AP22	U36.47

4.5 DDR Clocking

The DDR Clock is generated in the FPGA by using the Digital Clock Managers (DCM). Clocking for DDR SDRAM requires the transmission of two clocks, the

positive clock and the negative clock, SSTL_2 differential. These two clocks are 180° out of phase from each other, and their phase alignment must be tightly controlled. In order to prevent signal integrity problems and timing differences from becoming an issue, it is preferable for each device, whether memory or register, to have its own clock.

While it is possible for each device to have a positive and negative clock generated by the FPGA, this unnecessarily consumes pins that could be used elsewhere. To save these pins, an externally DDR SDRAM clock driver is used. The clock is routed to the DDR PLL Clock Driver that distributes the individual clocks to the separate DDR devices.

4.5.1 Clocking Methodology

This section describes the DDR clocking methodology implemented in the reference design (refer to [Figure 27](#)). The first DCM generates CLK0 and CLK90. CLK0 directly follows the user-supplied input clock (one of the clock sources, ACLK, BCLK etc.). This DCM also supplies the CLKDV output, which is the input clock divided by 16 used for the AUTO REFRESH counter. The second DCM in the controller block (DCM2_RECAPTURE) generates a phase-shifted version of the user input clock. It is used to recapture data from the DQS clock domain during a memory Read. Data recaptured in the rclk domain is then transferred to the system clock domain. The phase-shift value is specific to the system and must be programmed accordingly.

When adequate DCM resources are available, a third DCM can be used for better timing margins. This DCM is used to generate WCLK, a phase shifted version of the system clock. WCLK is used to clock data at the DDR IOB registers during a Write.

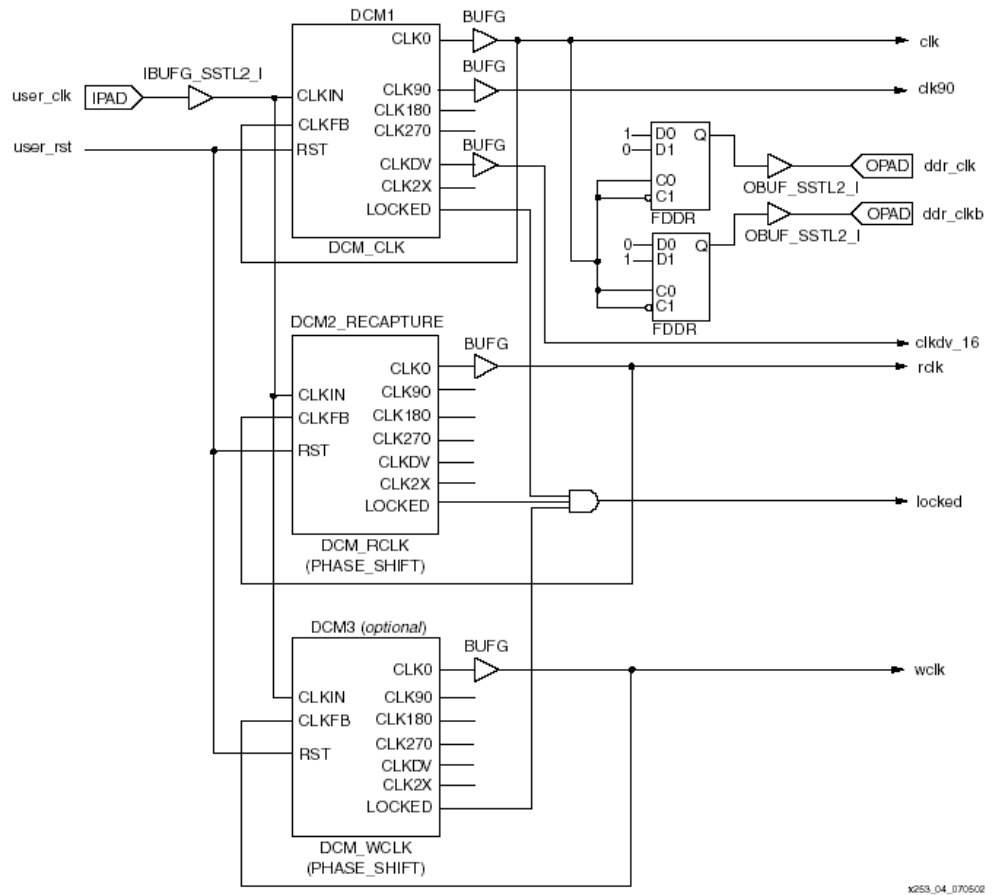


Figure 27 - DDR DCM Implementation

4.5.2 Connections between FPGA's and DDR PLL Clock Buffer

The connection between the FPGA's and the DDR PLL Clock Drivers consists of SSTL_2 differential pairs. A feedback reference clock input is provided from the PLL clock driver to each FPGA. The connections for all the FPGA's are shown in [Table 13](#).

Table 13 - Connection between FPGA's and DDR PLL Clock Drivers

Signal Name	FPGA Pin	DDR PLL Clock Driver
DDR_BCLKp	U14.AU22	U15.5
DDR_BCLKn	U14.AP22	U15.6
DDR_CCLKp	U31.K21	U32.5
DDR_CCLKn	U31.J21	U32.6
DDR_DCLKp	U35.AT21	U34.5

DDR_DCLKn	U35.AU21	U34.6
DDR_ECLKp	U55.K21	U53.5
DDR_ECLKn	U55.J21	U53.6
DDR_FCLKp	U54.AT21	U51.5
DDR_FCLKn	U54.U21	U51.6

4.6 Power PC (PPC) Clock – Sytem Clock

A 3.3 V half-can oscillator (X4), and the signal SYS_CLK provide an external clock source for the PPC. The oscillator is socketed and the DN6000K10PCI is shipped with a 100MHz oscillator, refer to [Figure 28](#).

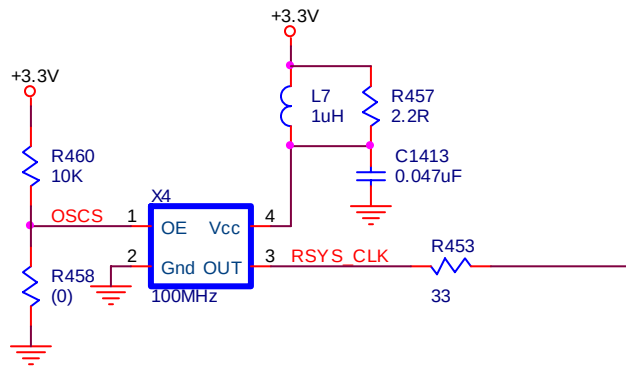


Figure 28 - PPC External Clock

4.6.1 Clocking Methodology

Refer to the Xilinx application notes for more information on this subject.

4.6.2 Connections between FPGA's and System Clock Buffer

The connection between the FPGA's and the external oscillator buffer are shown in [Table 14](#).

Table 14 - Connection between FPGA and External PPC Oscillator

Signal Name	FPGA Pin	DDR PLL Clock Driver
SYS_ACLKp	U14.AP21	U41.3
SYS_ACLKn	U14.AN21	U41.2
SYS_BCLKp	U16.K21	U41.5
SYS_BCLKn	U16.J21	U41.6
SYS_CCLKp	U31.AU22	U41.10

Signal Name	FPGA Pin	DDR PLL Clock Driver
SYS_CCLKn	U31.AT22	U41.9
SYS_DCLKp	U35.J22	U41.20
SYS_DCLKn	U35.K22	U41.19
SYS_ECLKp	U55.K21	U41.22
SYS_ECLKn	U55.J21	U41.23
SYS_FCLKp	U54.J22	U41.46
SYS_FCLKn	U54.K22	U41.47

4.7 Rocket IO Programmable Clocks

The DN6000K10PCI provides one crystal oscillator-to-differential LVDS frequency synthesizer per FPGA. These frequency synthesizers are serially programmable. The use of this variable clock source, allows designers to prototype various interconnect technologies with different clock source requirements. The dual output LVDS clocks are routed to the top and bottom RocketIO reference clock inputs. The PLL architecture for the RocketIO transceivers uses the reference clock as the interpolation source to clock the serial data. Removing the reference clock will stop the RX and TX PLLs from working. Therefore, a reference clock must be provided at all times. The serial transceiver input is locked to the input data stream through Clock and Data Recovery (CDR), a built in feature of the RocketIO transceiver. There are eight clock inputs into each RocketIO transceiver instantiation. REFCLK and BREFCLK are reference clocks generated from an external sources and presented to the FPGA as differential inputs. The reference clocks connect to the REFCLK or BREFCLK ports of the RocketIO multi-gigabit transceiver (MGT). While only one of these reference clocks is needed to drive the MGT, BREFCLK or BREFCLK2 must be used for serial speeds of 2.5 Gbps or greater. The reference clock also locks a Digital Clock Manager (DCM) or a BUFG to generate all of the other clocks for the GT. *Never run a reference clock through a DCM, since unwanted jitter will be introduced.*

4.7.1 Clocking Methodology

At speeds of 2.5 Gbps or greater, REFCLK configuration introduces more than the maximum allowable jitter to the RocketIO transceiver. For these higher speeds, BREFCLK configuration is required. The BREFCLK configuration uses dedicated routing resources that reduce jitter. BREFCLK must enter the FPGA through dedicated clock I/O. BREFCLK can connect to the BREFCLK inputs of the transceiver and the CLKIN input of the DCM for creation of USRCLKs. For more information refer to the Rocket IO User Guide available from the Xilinx website.

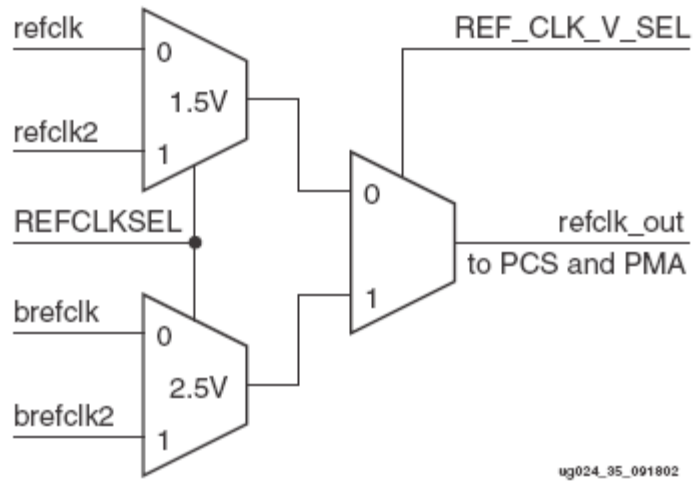


Figure 29 - REFCLK/BREFCLK Selection Logic

4.7.2 ICS8442 Programmable LVDS Clock Synthesizer

The DN6000K10PCI uses the ICS8442 LVDS clock synthesizer for generating various clock frequencies:

- VCO range: 250MHz to 700MHz
- Output Frequency range: 31.25MHz to 700MHz
- RMS period jitter: 2.7ps (typical)
- Cycle-to-cycle jitter: 18ps (typical)

Please refer to the manufacturers datasheet for more information

<http://www.icst.com/>

4.7.3 Connections between FPGA's and RocketIO Clock Synthesizers

The connection between the FPGA's and the RocketIO clock synthesizers are shown in [Table 15](#).

Table 15 - Connections between FPGA's and Rocket IO Clock Synthesizers

Signal Name	OSCILLATOR	FPGA Pin
RCKTIO_OSCT_Ap	U20.14	U14.F21
RCKTIO_OSCT_An	U20.15	U14.G21
RCKTIO_OSCB_Ap	U20.11	U14.AT21

Signal Name	OSCILLATOR	FPGA Pin
RCKTIO_OSCB_An	U20.12	U14.AU21
RCKTIO_OSCT_Bp	U22.14	U16.F21
RCKTIO_OSCT_Bn	U22.15	U16.G21
RCKTIO_OSCB_Bp	U22.11	U16.AT21
RCKTIO_OSCB_Bn	U22.12	U16.AU21
RCKTIO_OSCT_Cp	U45.14	U31.F22
RCKTIO_OSCT_Cn	U45.15	U31.G22
RCKTIO_OSCB_Cp	U45.11	U31.AT21
RCKTIO_OSCB_Cn	U45.12	U31.AU21
RCKTIO_OSCT_Dp	U27.14	U35.G22
RCKTIO_OSCT_Dn	U27.15	U35.F22
RCKTIO_OSCB_Dp	U27.11	U35.AU22
RCKTIO_OSCB_Dn	U27.12	U35.AT22
RCKTIO_OSCT_Ep	U56.14	U55.F21
RCKTIO_OSCT_En	U56.15	U55.G21
RCKTIO_OSCB_Ep	U56.11	U55.AT21
RCKTIO_OSCB_En	U56.12	U55.AU21
RCKTIO_OSCT_Fp	U59.14	U54.G22
RCKTIO_OSCT_Fn	U59.15	U54.F22
RCKTIO_OSCB_Fp	U59.11	U54.AU22
RCKTIO_OSCB_Fn	U59.12	U54.AT22

5 Reset Topology

5.1 DN6000K10PCI Reset

The voltage monitor device from Linear Technology, P/N LTC2900 (U1), allows a push-button reset function that is used to reset the DN6000K10PCI. [Figure 30](#) shows the distribution of the reset signal SYS_RSTn. In addition to controlling the reset, the power supplies rails +1.5V, +2.5V, +3.3V, and +5V are monitored for under-voltage conditions, that will cause the assertion of the SYS_RSTn signal. LED DS1.2 when lit, means that reset is asserted, refer the section describing the GPIO LED's.

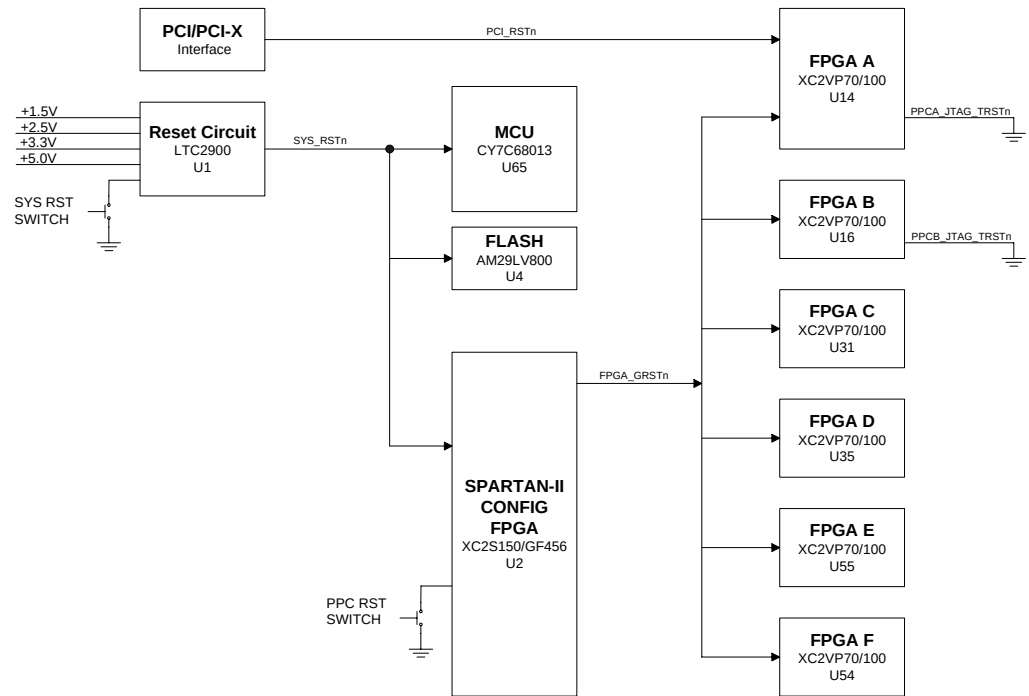


Figure 30 - Reset Topology Block Diagram

Depressing the reset push-button (S2) causes the following sequence of events:

1. Reset of the Configuration FPGA and MCU
2. Reset of FPGA's through FPGA_GRSTn signal
3. FPGA configuration is cleared
4. If the dipswitch is set for SelectMAP configuration option, and there is a valid SmartMedia card inserted into the socket, then the FPGA's will be configured. A SmartMedia card is valid if it complies with the SSFDC specification and contains a file named "main.txt" in the root directory. If the card is invalid or there is no card present, then the FPGA will not be configured.
5. The Main Menu will appear in the Terminal Window.

Note: The identical sequence of events occurs at power-up.

5.2 PPC Reset

The DN6000K10PCI also contains another RESET push-button (S4) used to reset the PPC's in each FPGA. This signal is pulled up on the DN6000K10PCI. The user is responsible for debouncing the reset signal in the Configuration FPGA. One of the MB[1..40] signals must be used to reset the PPC's in the FPGA's. Table 16 shows the connection between the reset push-button and the FPGA.

Table 16 - PPC Reset

Signal Name	FPGA Pin	Push-Button Switch
PPC_RESET _n	U2.H5	S3.4

6 Memory

The DN6000K10PCI provides two different memory technologies to the user. FLASH and DDR SDRAM in various densities.

6.1 Synchronous SRAM

The Synchronous SRAM memory components on the DN6000K10PCI can accommodate up to 2M x 36 devices, refer to [Figure 31](#) as an example of a SRAM interface (shown is the SRAM device on FPGA A).



- Pipelined
- Flow-through
- Pipelined with NoBL
- Flow-through with NoBL
- Pipelined ZBT

- Flow-through ZBT

Synburst Flow-through (Figure 32) is the most straightforward type of SSRAM. Write data may be accepted on the same clock cycle as the activation signal and address, and read data is returned one clock cycle after it is requested. Synburst is designed to allow two controllers to access the same SSRAM, using two activation signals, ADSC# and ADSP#; an activation with ADSP# requires data and byte enables one clock cycle after the address and activation.

Synburst Pipelined (Figure 33) is identical except for registered outputs, which delay read data an additional clock cycle but may be necessary for high speed designs.

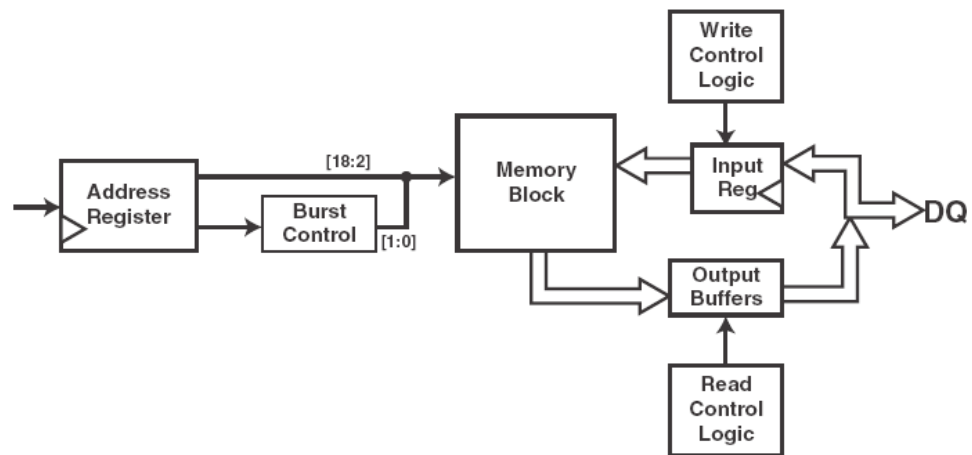


Figure 32 - SSRAM Flow-through

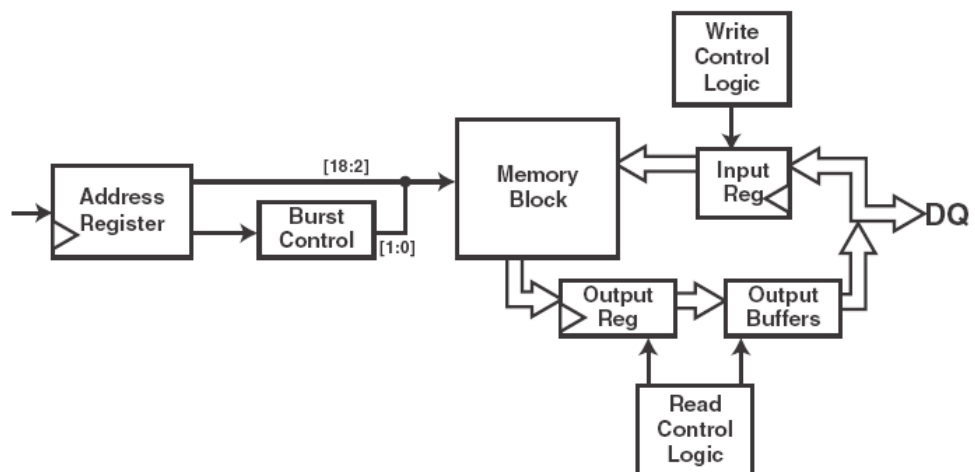


Figure 33 - SSRAM Pipeline

Zero-Bus-Turnaround (ZBT) SSRAM's are designed to eliminate wait states between reads and writes by synchronizing data. Figure 34 accept and return data one clock cycle after the address phase, and ZBT Pipeline SSRAMs (Figure 35) accept and return data two clock cycles after the address phase. This allows the user to begin a write burst immediately after the last word of a read burst, because read data will be returned before the first write data is required. The timing is illustrated in Figure 36.

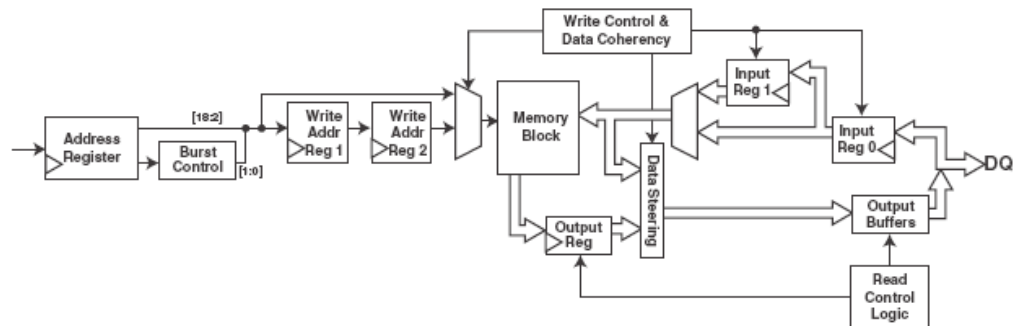


Figure 34 - SSRAM ZBT Flow-through

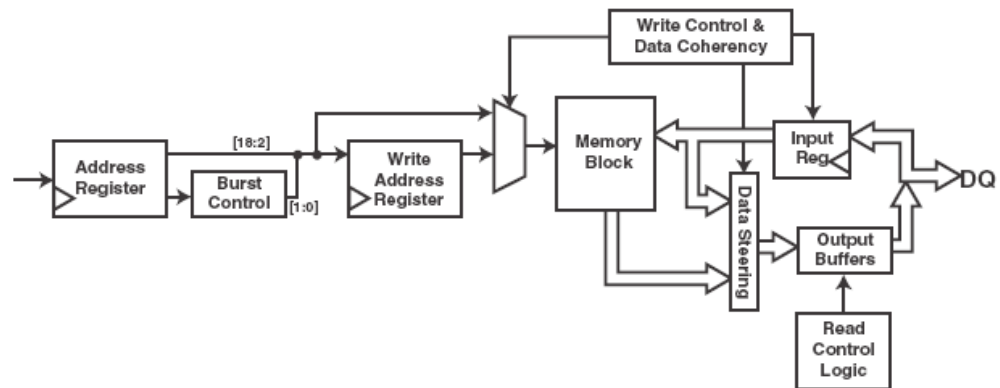


Figure 35 - SSRAM ZBT Pipeline

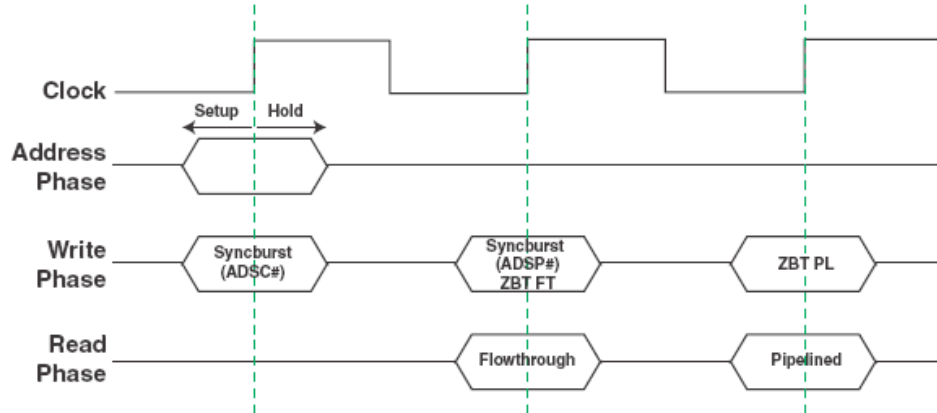


Figure 36 - Syncburst and ZBT SSRAM Timing

6.1.1 SSRAM Configuration

The DN6000K10PCI is factory stuffed with the Cypress P/N CY7C1380B-133AC SSRAM devices (please refer to datasheet for more information). There are 524,288 x 36 SSRAM cells with advanced synchronous peripheral circuitry and a 2-bit counter for internal burst operation. All synchronous inputs are gated by registers controlled by a positive-edge-triggered Clock Input (BCLK[6..9]). The synchronous inputs include all addresses, all data inputs, address-pipelining Chip Enable (CE), burst control inputs (ADSC, ADSP, and ADV), write enables (BWa, BWb, BWc, BWd and BWE), and Global Write (GW).

Asynchronous inputs include the Output Enable (OE) and burst mode control (MODE), DQa,b,c,d and DPa,b,c,d. a, b, c, d each are 8 bits wide in the case of DQ and 1 bit wide in the case of DP. Addresses and chip enables are registered with either Address Status Processor (ADSP) or Address Status Controller (ADSC) input pins. Subsequent burst addresses can be internally generated as controlled by the Burst Advance Pin (ADV).

Address, data inputs, and write controls are registered on-chip to initiate self-timed WRITE cycle. WRITE cycles can be one to four bytes wide as controlled by the write control inputs. Individual byte write allows individual byte to be written. Bwa controls DQa and DPa. BWb controls DQb and DPb. BWc controls DQc and DPc. BWd controls DQd and DPd. Bwa, BWb, BWc, and BWd can be active only with BWE being LOW. GW being LOW causes all bytes to be written. WRITE pass-through capability allows written data available at the output for the immediately next READ cycle. This device also incorporates pipelined enable circuit for easy depth expansion without penalizing system performance. All inputs and outputs of the CY7C1380B and is JEDEC standard JESD8-5 compatible.

Note: CE2 and CE2n are hard-wired on PWB to there respective active states. Use SRAM_CExn signal to select the individual devices.

6.1.2 SSRAM Clocking

The SSRAMs are clocked directly by RoboClock #2 (U9). BCLK6, BCLK7, BCLK8, and BCLK9 are LVTTTL33 signals and the SSRAMs are LVC MOS25. The CLK interface is level translated by the following circuit in Figure 37.

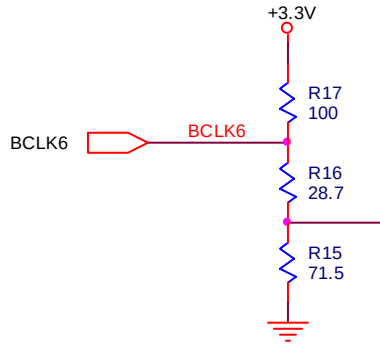


Figure 37 - Clock Level Translation

6.1.3 SRAM Termination

No termination is necessary, but the option to use DCI is available on all signals.

6.1.4 SRAM Connection to the FPGA's

Table 17 - Connection between FPGA and SRAM

Signal Name	SRAM Pin	FPGA Pin
FPGA A		
SRAM1_A0	U68.37	U14.V32
SRAM1_A1	U68.36	U14.V33
SRAM1_A2	U68.44	U14.U37
SRAM1_A3	U68.45	U14.U38
SRAM1_A4	U68.46	U14.U35
SRAM1_A5	U68.47	U14.U36
SRAM1_A6	U68.48	U14.T32

Signal Name	SRAM Pin	FPGA Pin
SRAM1_A7	U68.49	U14.T33
SRAM1_A8	U68.50	U14.T40
SRAM1_A9	U68.43	U14.T41
SRAM1_A10	U68.42	U14.T38
SRAM1_A11	U68.39	U14.T39
SRAM1_A12	U68.35	U14.U31
SRAM1_A13	U68.38	U14.R35
SRAM1_A14	U68.34	U14.T31
SRAM1_A15	U68.33	U14.U41
SRAM1_A16	U68.32	U14.U42
SRAM1_A17	U68.100	U14.U39
SRAM1_A18	U68.99	U14.U40
SRAM1_A19	U68.82	U14.U33
SRAM1_A20	U68.81	U14.U34
SRAM1_ADSCN	U68.85	U14.T37
SRAM1_ADSPN	U68.84	U14.T36
SRAM1_ADVN	U68.83	U14.T35
SRAM1_BWAN	U68.93	U14.R31
SRAM1_BWBN	U68.94	U14.R32

Signal Name	SRAM Pin	FPGA Pin
SRAM1_BWCN	U68.95	U14.R41
SRAM1_BWDN	U68.96	U14.R42
SRAM1_BWEN	U68.87	U14.R40
SRAM1_CE2	R13.2	U68.97
SRAM1_CE2N	R14.2	U68.92
SRAM1_CEN	U68.98	U14.R38
SRAM1_DQA0	U68.52	U14.AA39
SRAM1_DQA1	U68.53	U14.AA40
SRAM1_DQA2	U68.56	U14.AB31
SRAM1_DQA3	U68.57	U14.AA31
SRAM1_DQA4	U68.58	U14.AA36
SRAM1_DQA5	U68.59	U14.AA37
SRAM1_DQA6	U68.62	U14.AA33
SRAM1_DQA7	U68.63	U14.AA34
SRAM1_DQB0	U68.68	U14.Y31
SRAM1_DQB1	U68.69	U14.Y32
SRAM1_DQB2	U68.72	U14.Y39
SRAM1_DQB3	U68.73	U14.Y40
SRAM1_DQB4	U68.74	U14.Y36

Signal Name	SRAM Pin	FPGA Pin
SRAM1_DQB5	U68.75	U14.Y37
SRAM1_DQB6	U68.78	U14.Y33
SRAM1_DQB7	U68.79	U14.Y34
SRAM1_DQC0	U68.2	U14.W41
SRAM1_DQC1	U68.3	U14.W42
SRAM1_DQC2	U68.6	U14.W39
SRAM1_DQC3	U68.7	U14.W40
SRAM1_DQC4	U68.8	U14.W31
SRAM1_DQC5	U68.9	U14.W32
SRAM1_DQC6	U68.12	U14.W37
SRAM1_DQC7	U68.13	U14.W38
SRAM1_DQD0	U68.18	U14.W35
SRAM1_DQD1	U68.19	U14.W36
SRAM1_DQD2	U68.22	U14.W33
SRAM1_DQD3	U68.23	U14.W34
SRAM1_DQD4	U68.24	U14.V41
SRAM1_DQD5	U68.25	U14.V42
SRAM1_DQD6	U68.28	U14.V38
SRAM1_DQD7	U68.29	U14.V39

Signal Name	SRAM Pin	FPGA Pin
SRAM1_DQPA	U68.51	U14.V31
SRAM1_DQPB	U68.80	U14.U32
SRAM1_DQPC	U68.1	U14.V35
SRAM1_DQPD	U68.30	U14.V36
SRAM1_GWN	U68.88	U14.P40
SRAM1_LBON	U68.31	U14.R37
SRAM1_OEN	U68.86	U14.R33
SRAM1_ZZ	U68.64	U14.R34
FPGA B		
SRAM2_A0	U69.37	U16.AN35
SRAM2_A1	U69.36	U16.AN36
SRAM2_A2	U69.44	U16.AM38
SRAM2_A3	U69.45	U16.AM39
SRAM2_A4	U69.46	U16.AM34
SRAM2_A5	U69.47	U16.AM35
SRAM2_A6	U69.48	U16.AN40
SRAM2_A7	U69.49	U16.AM40
SRAM2_A8	U69.50	U16.AM41
SRAM2_A9	U69.43	U16.AM42

Signal Name	SRAM Pin	FPGA Pin
SRAM2_A10	U69.42	U16.AL33
SRAM2_A11	U69.39	U16.AL34
SRAM2_A12	U69.35	U16.AN37
SRAM2_A13	U69.38	U16.AL35
SRAM2_A14	U69.34	U16.AN38
SRAM2_A15	U69.33	U16.AN41
SRAM2_A16	U69.32	U16.AN42
SRAM2_A17	U69.100	U16.AM33
SRAM2_A18	U69.99	U16.AN34
SRAM2_A19	U69.82	U16.AM36
SRAM2_A20	U69.81	U16.AM37
SRAM2_ADSCN	U69.85	U16.AL39
SRAM2_ADSPN	U69.84	U16.AL38
SRAM2_ADVN	U69.83	U16.AL36
SRAM2_BWAN	U69.93	U16.AL31
SRAM2_BWBN	U69.94	U16.AL32
SRAM2_BWCN	U69.95	U16.AL40
SRAM2_BWDN	U69.96	U16.AL41
SRAM2_BWEN	U69.87	U16.AK35

Signal Name	SRAM Pin	FPGA Pin
SRAM2_CE2	R22.2	U69.97
SRAM2_CE2N	R21.2	U69.92
SRAM2_CEN	U69.98	U16.AK34
SRAM2_DQA0	U16.AW36	U69.52
SRAM2_DQA1	U16.AV36	U69.53
SRAM2_DQA2	U16.AY37	U69.56
SRAM2_DQA3	U16.AY38	U69.57
SRAM2_DQA4	U16.AU36	U69.58
SRAM2_DQA5	U16.AT37	U69.59
SRAM2_DQA6	U16.AU35	U69.62
SRAM2_DQA7	U16.AT35	U69.63
SRAM2_DQB0	U16.AW41	U69.68
SRAM2_DQB1	U16.AW42	U69.69
SRAM2_DQB2	U16.AV41	U69.72
SRAM2_DQB3	U16.AV42	U69.73
SRAM2_DQB4	U16.AW40	U69.74
SRAM2_DQB5	U16.AV40	U69.75
SRAM2_DQB6	U16.AU39	U69.78
SRAM2_DQB7	U16.AU40	U69.79

Signal Name	SRAM Pin	FPGA Pin
SRAM2_DQC0	U16.AU41	U69.2
SRAM2_DQC1	U16.AU42	U69.3
SRAM2_DQC2	U16.AT39	U69.6
SRAM2_DQC3	U16.AT40	U69.7
SRAM2_DQC4	U16.AT41	U69.8
SRAM2_DQC5	U16.AT42	U69.9
SRAM2_DQC6	U16.AR38	U69.12
SRAM2_DQC7	U16.AR39	U69.13
SRAM2_DQD0	U16.AR37	U69.18
SRAM2_DQD1	U16.AT38	U69.19
SRAM2_DQD2	U16.AR40	U69.22
SRAM2_DQD3	U16.AR41	U69.23
SRAM2_DQD4	U16.AP36	U69.24
SRAM2_DQD5	U16.AP37	U69.25
SRAM2_DQD6	U16.AP35	U69.28
SRAM2_DQD7	U16.AR36	U69.29
SRAM2_DQPA	U16.AP38	U69.51
SRAM2_DQPB	U16.AP39	U69.80
SRAM2_DQPC	U16.AP41	U69.1

Signal Name	SRAM Pin	FPGA Pin
SRAM2_DQPD	U16.AP42	U69.30
SRAM2_GWN	U69.88	U16.AK36
SRAM2_LBON	U69.31	U16.AK33
SRAM2_OEN	U69.86	U16.AK37
SRAM2_ZZ	U69.64	U16.AK38
FPGA E		
SRAM3_A0	U70.37	U54.AF1
SRAM3_A1	U70.36	U54.AF2
SRAM3_A2	U70.44	U54.AF9
SRAM3_A3	U70.45	U54.AF10
SRAM3_A4	U70.46	U54.AG2
SRAM3_A5	U70.47	U54.AG3
SRAM3_A6	U70.48	U54.AG10
SRAM3_A7	U70.49	U54.AG11
SRAM3_A8	U70.50	U54.AG4
SRAM3_A9	U70.43	U54.AG5
SRAM3_A10	U70.42	U54.AG6
SRAM3_A11	U70.39	U54.AG7
SRAM3_A12	U70.35	U54.AG12

Signal Name	SRAM Pin	FPGA Pin
SRAM3_A13	U70.38	U54.AG8
SRAM3_A14	U70.34	U54.AF12
SRAM3_A15	U70.33	U54.AF3
SRAM3_A16	U70.32	U54.AF4
SRAM3_A17	U70.100	U54.AF5
SRAM3_A18	U70.99	U54.AF6
SRAM3_A19	U70.82	U54.AF7
SRAM3_A20	U70.81	U54.AF8
SRAM3_ADSCN	U70.85	U54.AH2
SRAM3_ADSPN	U70.84	U54.AH1
SRAM3_ADVN	U70.83	U54.AH8
SRAM3_BWAN	U70.93	U54.AH3
SRAM3_BWBN	U70.94	U54.AJ3
SRAM3_BWCN	U70.95	U54.AH11
SRAM3_BWDN	U70.96	U54.AH12
SRAM3_BWEN	U70.87	U54.AH5
SRAM3_CE2	U70.97	R82.2
SRAM3_CE2N	U70.92	R81.2
SRAM3_CEN	U70.98	U54.AH10

Signal Name	SRAM Pin	FPGA Pin
SRAM3_DQA0	U70.52	U54.AB3
SRAM3_DQA1	U70.53	U54.AB4
SRAM3_DQA2	U70.56	U54.AB6
SRAM3_DQA3	U70.57	U54.AB7
SRAM3_DQA4	U70.58	U54.AB9
SRAM3_DQA5	U70.59	U54.AB10
SRAM3_DQA6	U70.62	U54.AC3
SRAM3_DQA7	U70.63	U54.AC4
SRAM3_DQB0	U70.68	U54.AC11
SRAM3_DQB1	U70.69	U54.AC12
SRAM3_DQB2	U70.72	U54.AC6
SRAM3_DQB3	U70.73	U54.AC7
SRAM3_DQB4	U70.74	U54.AC9
SRAM3_DQB5	U70.75	U54.AC10
SRAM3_DQB6	U70.78	U54.AD9
SRAM3_DQB7	U70.79	U54.AD10
SRAM3_DQC0	U70.2	U54.AD1
SRAM3_DQC1	U70.3	U54.AD2
SRAM3_DQC2	U70.6	U54.AD3

Signal Name	SRAM Pin	FPGA Pin
SRAM3_DQC3	U70.7	U54.AD4
SRAM3_DQC4	U70.8	U54.AD11
SRAM3_DQC5	U70.9	U54.AD12
SRAM3_DQC6	U70.12	U54.AD5
SRAM3_DQC7	U70.13	U54.AD6
SRAM3_DQD0	U70.18	U54.AD7
SRAM3_DQD1	U70.19	U54.AD8
SRAM3_DQD2	U70.22	U54.AE10
SRAM3_DQD3	U70.23	U54.AE11
SRAM3_DQD4	U70.24	U54.AE1
SRAM3_DQD5	U70.25	U54.AE2
SRAM3_DQD6	U70.28	U54.AE4
SRAM3_DQD7	U70.29	U54.AE5
SRAM3_DQPA	U70.51	U54.AF11
SRAM3_DQPB	U70.80	U54.AE12
SRAM3_DQPC	U70.1	U54.AE7
SRAM3_DQPD	U70.30	U54.AE8
SRAM3_GWN	U70.88	U54.AH6
SRAM3_LBON	U70.31	U54.AH9

Signal Name	SRAM Pin	FPGA Pin
SRAM3_OEN	U70.86	U54.AJ11
SRAM3_ZZ	U70.64	U54.AJ12
FPGA F		
SRAM4_A0	U71.37	U55.K6
SRAM4_A1	U71.36	U55.K5
SRAM4_A2	U71.44	U55.K3
SRAM4_A3	U71.45	U55.L3
SRAM4_A4	U71.46	U55.L5
SRAM4_A5	U71.47	U55.L4
SRAM4_A6	U71.48	U55.L1
SRAM4_A7	U71.49	U55.L2
SRAM4_A8	U71.50	U55.M7
SRAM4_A9	U71.43	U55.M8
SRAM4_A10	U71.42	U55.M11
SRAM4_A11	U71.39	U55.M12
SRAM4_A12	U71.35	U55.K8
SRAM4_A13	U71.38	U55.M9
SRAM4_A14	U71.34	U55.K7
SRAM4_A15	U71.33	U55.K2

Signal Name	SRAM Pin	FPGA Pin
SRAM4_A16	U71.32	U55.K1
SRAM4_A17	U71.100	U55.L8
SRAM4_A18	U71.99	U55.L9
SRAM4_A19	U71.82	U55.L6
SRAM4_A20	U71.81	U55.L7
SRAM4_ADSCN	U71.85	U55.M3
SRAM4_ADSPN	U71.84	U55.M2
SRAM4_ADVN	U71.83	U55.M10
SRAM4_BWAN	U71.93	U55.M4
SRAM4_BWBN	U71.94	U55.M5
SRAM4_BWCN	U71.95	U55.N7
SRAM4_BWDN	U71.96	U55.N8
SRAM4_BWEN	U71.87	U55.N5
SRAM4_CE2	U71.97	R73.2
SRAM4_CE2N	U71.92	R74.2
SRAM4_CEN	U71.98	U55.N10
SRAM4_DQA0	U71.52	U55.E7
SRAM4_DQA1	U71.53	U55.D7
SRAM4_DQA2	U71.56	U55.E6

Signal Name	SRAM Pin	FPGA Pin
SRAM4_DQA3	U71.57	U55.D6
SRAM4_DQA4	U71.58	U55.G6
SRAM4_DQA5	U71.59	U55.F7
SRAM4_DQA6	U71.62	U55.D3
SRAM4_DQA7	U71.63	U55.E3
SRAM4_DQB0	U71.68	U55.D1
SRAM4_DQB1	U71.69	U55.D2
SRAM4_DQB2	U71.72	U55.E1
SRAM4_DQB3	U71.73	U55.E2
SRAM4_DQB4	U71.74	U55.F4
SRAM4_DQB5	U71.75	U55.F3
SRAM4_DQB6	U71.78	U55.F1
SRAM4_DQB7	U71.79	U55.F2
SRAM4_DQC0	U71.2	U55.G3
SRAM4_DQC1	U71.3	U55.G4
SRAM4_DQC2	U71.6	U55.G2
SRAM4_DQC3	U71.7	U55.G1
SRAM4_DQC4	U71.8	U55.G5
SRAM4_DQC5	U71.9	U55.H6

Signal Name	SRAM Pin	FPGA Pin
SRAM4_DQC6	U71.12	U55.H4
SRAM4_DQC7	U71.13	U55.H5
SRAM4_DQD0	U71.18	U55.H3
SRAM4_DQD1	U71.19	U55.H2
SRAM4_DQD2	U71.22	U55.H7
SRAM4_DQD3	U71.23	U55.J8
SRAM4_DQD4	U71.24	U55.J6
SRAM4_DQD5	U71.25	U55.J7
SRAM4_DQD6	U71.28	U55.J5
SRAM4_DQD7	U71.29	U55.J4
SRAM4_DQPA	U71.51	U55.J1
SRAM4_DQPB	U71.80	U55.J2
SRAM4_DQPC	U71.1	U55.K9
SRAM4_DQPD	U71.30	U55.L10
SRAM4_GWN	U71.88	U55.N6
SRAM4_LBON	U71.31	U55.N9
SRAM4_OEN	U71.86	U55.N3
SRAM4_ZZ	U71.64	U55.N4
SRAM_CSN	U3.5	U2.E21

6.2 DDR SDRAM

Double Data Rate (DDR) SDRAM represents an enhancement to the traditional SDRAM. Instead of data and control signals operating at the same frequency, data operates at twice the clock frequency, while address and control operate at the base clock frequency. In other words, the data is written or read from the device on every clock transition, or twice per clock cycle. This effectively doubles the throughput of the memory device.

The trade-off for such an improvement in throughput is increased complexity in interface logic to the DDR memory, as well as increased complexity in routing the DDR signals on the printed circuit board. Additionally, this memory has the same latencies as standard SDRAM, so that while the data transfers are twice as fast, the latencies associated with DDR SDRAM are on par with standard SDRAM.

6.2.1 Basics of DDR Operation

DDR SDRAM provides data capture at a rate of twice the clock frequency. Therefore, DDR SDRAM with a clock frequency of 100 MHz has a peak data transfer rate of 200 MHz or 6.4 Gigabits per second for a 16-bit interface. In order to maintain high-speed signal integrity and stringent timing goals, a bi-directional data strobe is used in conjunction with SSTL_2 signaling standard as well as differential clocks. DDR SDRAM operates as a source-synchronous system, in which data is captured twice per clock cycle, using a bi-directional data strobe to clock the data. The DDR SDRAM control bus consists of a clock enable, chip select, row and column addresses, bank address, and a write enable. Commands are entered on the positive edges of the clock, and data occurs for both positive and negative edges of the clock. The double data rate memory utilizes a differential pair for the system clock and, therefore, has both a true clock (CKp) and complementary clock (CKn) signal.

6.2.2 DDR SDRAM Configuration

The DDR SDRAM memory components on the DN6000K10PCI are arranged as a 16-bit mode, refer to [Figure 38](#) as an example of a DDR interface (shown is the DRR device on FPGA B). Each FPGA has two discrete parts (U22, U32 etc). The components can be up to 64Mb x 16 parts, organized as 16 million deep by 16-bits wide and 4 banks (for more information, refer to Micron's datasheet PN MT46V64M16).

SSTL2 Class 1 termination is used for unidirectional signaling, such as control signals. It is based on a 50Ω controlled impedance driver, a 50Ω controlled impedance transmission line, and a 50Ω parallel termination to V_{TT} at the receiver. Figure 39 shows a basic SSTL2 Class 1 circuit. The driver is brought to 50Ω by the addition of a 25Ω series resistor immediately adjacent to the driver (implemented using DCI, thus no need for an external component).

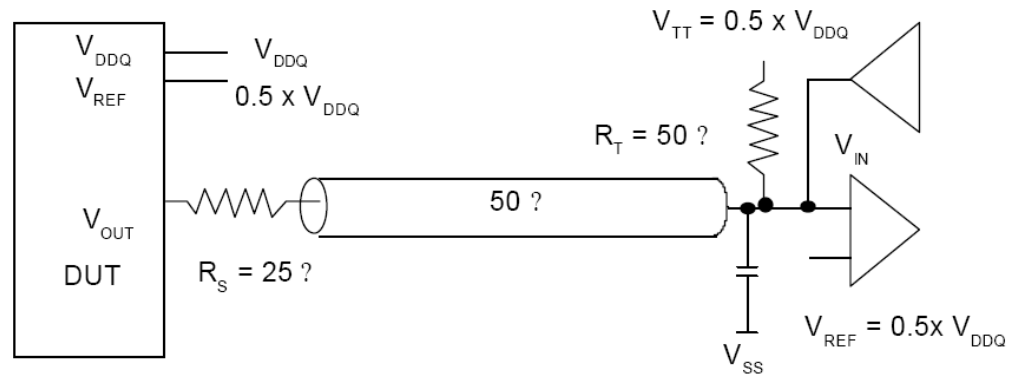


Figure 39 - SSTL2 Class 1 Termination

SSTL2 Class 2 termination is used for bi-directional signaling, such as data signals. It is based on a 50Ω controlled impedance driver and a 50Ω parallel termination to V_{TT} for the receiver at both ends, connected through a 50Ω controlled impedance transmission line. Figure 40 shows a basic SSTL2 Class 2 circuit. The driver is brought to 50Ω by the addition of a 25Ω series resistor immediately adjacent to the driver.

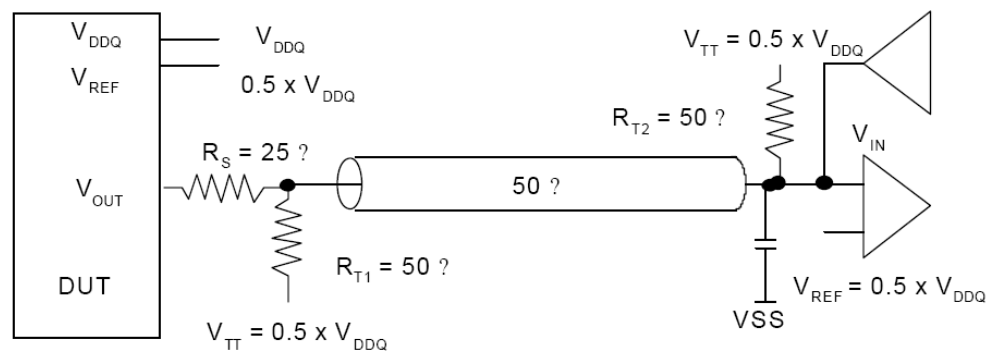
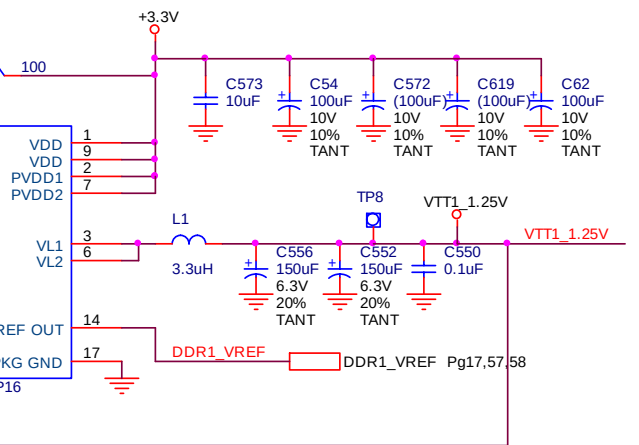


Figure 40 - SSTL2 Class 2 Termination

Note: DCI termination must be implemented in the DDR SDRAM controller design.

ply

used to supply power to the +2.5V plane that supplies DRAM devices. Due to the power requirements, three supply pins are used to supply the power to the DRR devices on FPGA B, C/D, and E. The JEDEC Specification – Double Data Rate (DDR) Type 2 (JEDEC JESD79-1) VTT must track 50% of VDDQ over voltage, and the VREF pin (U8, U26, U46) is used as a voltage source for the VREF pin to the +2.5V supply allows the regulator to maintain the VREF pin (Figure 41). A dedicated VREF output supplies the voltage to the DDR SDRAM devices and maintains a less



41 - DDR VTT Termination Regulator

Access to the FPGA

GA and the DDR SDRAM are not homogeneous, as 1 and differently from the data and differently from the signals are controlled impedance, and are SSTL2 these signals is covered in [DDR SDRAM Termination](#).

a Strobe (DQS) and the Data Mask (DM) signals are in the FPGA to the DDR SDRAM components. As are controlled impedance, and terminated according to The data, data strobe, and data mask signals all serve als are self-evident, carrying the raw data between the data strobe signals are responsible for actual clocking dges of the clock. Finally, the data mask signals can be ding and writing of some of the bytes in a 16-bit word als between the FPGA and the DDR SDRAM

Table 18 - Connection between FPGA's and DDR SDRAM's

Signal Name	FPGA Pin	DDR SDRAM
DDR_B1_DATA0	U16.AW33	U11.2
DDR_B1_DATA1	U16.AV33	U11.4
DDR_B1_DATA10	U16.AN30	U11.57
DDR_B1_DATA11	U16.AP30	U11.59
DDR_B1_DATA12	U16.AL30	U11.60
DDR_B1_DATA13	U16.AM30	U11.62
DDR_B1_DATA14	U16.AR30	U11.63
DDR_B1_DATA15	U16.AT30	U11.65
DDR_B1_DATA2	U16.AY32	U11.5
DDR_B1_DATA3	U16.AY33	U11.7
DDR_B1_DATA4	U16.AU32	U11.8
DDR_B1_DATA5	U16.AV32	U11.10
DDR_B1_DATA6	U16.AM31	U11.11
DDR_B1_DATA7	U16.AN31	U11.13
DDR_B1_DATA8	U16.AU31	U11.54
DDR_B1_DATA9	U16.AT31	U11.56
DDR_B1_ADD0	U16.AY30	U11.29
DDR_B1_ADD1	U16.AW30	U11.30
DDR_B1_ADD10	U16.AN27	U11.28

Signal Name	FPGA Pin	DDR SDRAM
DDR_B1_ADD11	U16.AP27	U11.41
DDR_B1_ADD12	U16.AN26	U11.42
DDR_B1_ADD13	U16.AM26	U11.17
DDR_B1_ADD2	U16.AU30	U11.31
DDR_B1_ADD3	U16.AV30	U11.32
DDR_B1_ADD4	U16.AU28	U11.35
DDR_B1_ADD5	U16.AV28	U11.36
DDR_B1_ADD6	U16.AL27	U11.37
DDR_B1_ADD7	U16.AM27	U11.38
DDR_B1_ADD8	U16.AT27	U11.39
DDR_B1_ADD9	U16.AR27	U11.40
DDR_B1_BA0	U16.AM23	U11.26
DDR_B1_BA1	U16.AN23	U11.27
DDR_B1_CASN	U16.AP23	U11.22
DDR_B1_CKE	U16.AR32	U11.44
DDR_B1_CSN	U16.AL28	U11.24
DDR_B1_RASN	U16.AR23	U11.23
DDR_B1_LDM	U16.AU33	U11.20
DDR_B1_LDQS	U16.AN32	U11.16
DDR_B1_UDM	U16.AW31	U11.47

Signal Name	FPGA Pin	DDR SDRAM
DDR_B1_UDQS	U16.AP31	U11.51
DDR_B1_WEN	U16.AV27	U11.21
DDR_B2_DATA0	U16.AT29	U19.2
DDR_B2_DATA1	U16.AU29	U19.4
DDR_B2_DATA10	U16.AV25	U19.57
DDR_B2_DATA11	U16.AV26	U19.59
DDR_B2_DATA12	U16.AR25	U19.60
DDR_B2_DATA13	U16.AT25	U19.62
DDR_B2_DATA14	U16.AN24	U19.63
DDR_B2_DATA15	U16.AP24	U19.65
DDR_B2_DATA2	U16.AN28	U19.5
DDR_B2_DATA3	U16.AM28	U19.7
DDR_B2_DATA4	U16.AV29	U19.8
DDR_B2_DATA5	U16.AW29	U19.10
DDR_B2_DATA6	U16.AY28	U19.11
DDR_B2_DATA7	U16.AY29	U19.13
DDR_B2_DATA8	U16.AM25	U19.54
DDR_B2_DATA9	U16.AN25	U19.56
DDR_B2_ADD0	U16.AL26	U19.29
DDR_B2_ADD1	U16.AL25	U19.30

Signal Name	FPGA Pin	DDR SDRAM
DDR_B2_ADD10	U16.AY24	U19.28
DDR_B2_ADD11	U16.AW24	U19.41
DDR_B2_ADD12	U16.AU24	U19.42
DDR_B2_ADD13	U16.AV24	U19.17
DDR_B2_ADD2	U16.AP26	U19.31
DDR_B2_ADD3	U16.AR26	U19.32
DDR_B2_ADD4	U16.AT26	U19.35
DDR_B2_ADD5	U16.AU26	U19.36
DDR_B2_ADD6	U16.AL24	U19.37
DDR_B2_ADD7	U16.AM24	U19.38
DDR_B2_ADD8	U16.AR24	U19.39
DDR_B2_ADD9	U16.AT24	U19.40
DDR_B2_LDM	U16.AR28	U19.20
DDR_B2_LDQS	U16.AR29	U19.16
DDR_B2_UDM	U16.AY25	U19.47
DDR_B2_UDQS	U16.AW26	U19.51
DDR_B2_BA0	U16.AN29	U19.26
DDR_B2_BA1	U16.AM29	U19.27
DDR_B2_CASN	U16.AY23	U19.22
DDR_B2_CKE	U16.AW27	U19.44

Signal Name	FPGA Pin	DDR SDRAM
DDR_B2_CSN	U16.AL22	U19.24
DDR_B2_RASN	U16.AW23	U19.23
DDR_B2_WEN	U16.AV23	U19.21
DDR_C1_DATA0	U31.M18	U28.2
DDR_C1_DATA1	U31.M17	U28.4
DDR_C1_DATA10	U31.G18	U28.57
DDR_C1_DATA11	U31.H18	U28.59
DDR_C1_DATA12	U31.E17	U28.60
DDR_C1_DATA13	U31.E18	U28.62
DDR_C1_DATA14	U31.J19	U28.63
DDR_C1_DATA15	U31.K19	U28.65
DDR_C1_DATA2	U31.L17	U28.5
DDR_C1_DATA3	U31.K17	U28.7
DDR_C1_DATA4	U31.H17	U28.8
DDR_C1_DATA5	U31.J17	U28.10
DDR_C1_DATA6	U31.F17	U28.11
DDR_C1_DATA7	U31.G17	U28.13
DDR_C1_DATA8	U31.K18	U28.54
DDR_C1_DATA9	U31.L18	U28.56
DDR_C1_ADD0	U31.C14	U28.29

Signal Name	FPGA Pin	DDR SDRAM
DDR_C1_ADD1	U31.C15	U28.30
DDR_C1_ADD10	U31.E19	U28.28
DDR_C1_ADD11	U31.F19	U28.41
DDR_C1_ADD12	U31.D19	U28.42
DDR_C1_ADD13	U31.C19	U28.17
DDR_C1_ADD2	U31.L16	U28.31
DDR_C1_ADD3	U31.M16	U28.32
DDR_C1_ADD4	U31.J16	U28.35
DDR_C1_ADD5	U31.K16	U28.36
DDR_C1_ADD6	U31.H16	U28.37
DDR_C1_ADD7	U31.G16	U28.38
DDR_C1_ADD8	U31.G19	U28.39
DDR_C1_ADD9	U31.H19	U28.40
DDR_C1_BA0	U31.C20	U28.26
DDR_C1_BA1	U31.D20	U28.27
DDR_C1_CASN	U31.K20	U28.22
DDR_C1_CKE	U31.M21	U28.44
DDR_C1_CSN	U31.J20	U28.24
DDR_C1_LDM	U31.D16	U28.20
DDR_C1_LDQS	U31.D17	U28.16

Signal Name	FPGA Pin	DDR SDRAM
DDR_C1_RASN	U31.L20	U28.23
DDR_C1_UDM	U31.C18	U28.47
DDR_C1_UDQS	U31.M19	U28.51
DDR_C1_WEN	U31.H20	U28.21
DDR_C2_DATA0	U31.H10	U37.2
DDR_C2_DATA1	U31.J10	U37.4
DDR_C2_DATA10	U31.F14	U37.57
DDR_C2_DATA11	U31.G14	U37.59
DDR_C2_DATA12	U31.D14	U37.60
DDR_C2_DATA13	U31.E14	U37.62
DDR_C2_DATA14	U31.L15	U37.63
DDR_C2_DATA15	U31.K15	U37.65
DDR_C2_DATA2	U31.F10	U37.5
DDR_C2_DATA3	U31.G10	U37.7
DDR_C2_DATA4	U31.E10	U37.8
DDR_C2_DATA5	U31.D10	U37.10
DDR_C2_DATA6	U31.C10	U37.11
DDR_C2_DATA7	U31.C11	U37.13
DDR_C2_DATA8	U31.K14	U37.54
DDR_C2_DATA9	U31.L14	U37.56

Signal Name	FPGA Pin	DDR SDRAM
DDR_C2_ADD0	U31.G12	U37.29
DDR_C2_ADD1	U31.F12	U37.30
DDR_C2_ADD10	U31.F13	U37.28
DDR_C2_ADD11	U31.D13	U37.41
DDR_C2_ADD12	U31.C13	U37.42
DDR_C2_ADD13	U31.M15	U37.17
DDR_C2_ADD2	U31.D12	U37.31
DDR_C2_ADD3	U31.L13	U37.32
DDR_C2_ADD4	U31.M13	U37.35
DDR_C2_ADD5	U31.J13	U37.36
DDR_C2_ADD6	U31.K13	U37.37
DDR_C2_ADD7	U31.G13	U37.38
DDR_C2_ADD8	U31.H13	U37.39
DDR_C2_ADD9	U31.E13	U37.40
DDR_C2_BA0	U31.F9	U37.26
DDR_C2_BA1	U31.E9	U37.27
DDR_C2_CASN	U31.H12	U37.22
DDR_C2_CKE	U31.K11	U37.44
DDR_C2_CSN	U31.K12	U37.24
DDR_C2_LDM	U31.H11	U37.20

Signal Name	FPGA Pin	DDR SDRAM
DDR_C2_LDQS	U31.F11	U37.16
DDR_C2_RASN	U31.J12	U37.23
DDR_C2_UDM	U31.H15	U37.47
DDR_C2_UDQS	U31.F15	U37.51
DDR_C2_WEN	U31.L12	U37.21
DDR_D1_DATA0	U35.AN20	U29.2
DDR_D1_DATA1	U35.AM20	U29.4
DDR_D1_DATA10	U35.AV17	U29.57
DDR_D1_DATA11	U35.AV18	U29.59
DDR_D1_DATA12	U35.AN18	U29.60
DDR_D1_DATA13	U35.AM18	U29.62
DDR_D1_DATA14	U35.AU17	U29.63
DDR_D1_DATA15	U35.AT17	U29.65
DDR_D1_DATA2	U35.AP20	U29.5
DDR_D1_DATA3	U35.AR20	U29.7
DDR_D1_DATA4	U35.AV19	U29.8
DDR_D1_DATA5	U35.AU19	U29.10
DDR_D1_DATA6	U35.AW19	U29.11
DDR_D1_DATA7	U35.AY19	U29.13
DDR_D1_DATA8	U35.AT18	U29.54

Signal Name	FPGA Pin	DDR SDRAM
DDR_D1_DATA9	U35.AR18	U29.56
DDR_D1_ADD0	U35.AT19	U29.29
DDR_D1_ADD1	U35.AR19	U29.30
DDR_D1_ADD10	U35.AM17	U29.28
DDR_D1_ADD11	U35.AN17	U29.41
DDR_D1_ADD12	U35.AP16	U29.42
DDR_D1_ADD13	U35.AN16	U29.17
DDR_D1_ADD2	U35.AM19	U29.31
DDR_D1_ADD3	U35.AL19	U29.32
DDR_D1_ADD4	U35.AP19	U29.35
DDR_D1_ADD5	U35.AN19	U29.36
DDR_D1_ADD6	U35.AR17	U29.37
DDR_D1_ADD7	U35.AP17	U29.38
DDR_D1_ADD8	U35.AL18	U29.39
DDR_D1_ADD9	U35.AL17	U29.40
DDR_D1_LDM	U35.AL21	U29.20
DDR_D1_LDQS	U35.AV20	U29.16
DDR_D1_UDM	U35.AW17	U29.47
DDR_D1_UDQS	U35.AY18	U29.51
DDR_D1_BA0	U35.AW20	U29.26

Signal Name	FPGA Pin	DDR SDRAM
DDR_D1_BA1	U35.AY20	U29.27
DDR_D1_CASN	U35.AT16	U29.22
DDR_D1_CKE	U35.AW16	U29.44
DDR_D1_CSN	U35.AV16	U29.24
DDR_D1_RASN	U35.AR16	U29.23
DDR_D1_WEN	U35.AL15	U29.21
DDR_D2_DATA0	U35.AW14	U38.2
DDR_D2_DATA1	U35.AV14	U38.4
DDR_D2_DATA10	U35.AP13	U38.57
DDR_D2_DATA11	U35.AN13	U38.59
DDR_D2_DATA12	U35.AR12	U38.60
DDR_D2_DATA13	U35.AP12	U38.62
DDR_D2_DATA14	U35.AT12	U38.63
DDR_D2_DATA15	U35.AU12	U38.65
DDR_D2_DATA2	U35.AM15	U38.5
DDR_D2_DATA3	U35.AN15	U38.7
DDR_D2_DATA4	U35.AU14	U38.8
DDR_D2_DATA5	U35.AT14	U38.10
DDR_D2_DATA6	U35.AN14	U38.11
DDR_D2_DATA7	U35.AM14	U38.13

Signal Name	FPGA Pin	DDR SDRAM
DDR_D2_DATA8	U35.AM13	U38.54
DDR_D2_DATA9	U35.AL13	U38.56
DDR_D2_ADD0	U35.AV15	U38.29
DDR_D2_ADD1	U35.AU15	U38.30
DDR_D2_ADD10	U35.AV11	U38.28
DDR_D2_ADD11	U35.AU11	U38.41
DDR_D2_ADD12	U35.AY10	U38.42
DDR_D2_ADD13	U35.AY11	U38.17
DDR_D2_ADD2	U35.AY14	U38.31
DDR_D2_ADD3	U35.AY15	U38.32
DDR_D2_ADD4	U35.AV13	U38.35
DDR_D2_ADD5	U35.AU13	U38.36
DDR_D2_ADD6	U35.AW13	U38.37
DDR_D2_ADD7	U35.AY13	U38.38
DDR_D2_ADD8	U35.AN12	U38.39
DDR_D2_ADD9	U35.AM12	U38.40
DDR_D2_LDM	U35.AR14	U38.20
DDR_D2_LDQS	U35.AR15	U38.16
DDR_D2_UDM	U35.AW12	U38.47
DDR_D2_UDQS	U35.AT13	U38.51

Signal Name	FPGA Pin	DDR SDRAM
DDR_D2_BA0	U35.AL16	U38.26
DDR_D2_BA1	U35.AM16	U38.27
DDR_D2_CASN	U35.AW10	U38.22
DDR_D2_CKE	U35.AN11	U38.44
DDR_D2_CSN	U35.AR11	U38.24
DDR_D2_RASN	U35.AV10	U38.23
DDR_D2_WEN	U35.AU10	
DDR_E1_DATA0	U55.M18	U49.2
DDR_E1_DATA1	U55.M17	U49.4
DDR_E1_DATA10	U55.G18	U49.57
DDR_E1_DATA11	U55.H18	U49.59
DDR_E1_DATA12	U55.E17	U49.60
DDR_E1_DATA13	U55.E18	U49.62
DDR_E1_DATA14	U55.J19	U49.63
DDR_E1_DATA15	U55.K19	U49.65
DDR_E1_DATA2	U55.L17	U49.5
DDR_E1_DATA3	U55.K17	U49.7
DDR_E1_DATA4	U55.H17	U49.8
DDR_E1_DATA5	U55.J17	U49.10
DDR_E1_DATA6	U55.F17	U49.11

Signal Name	FPGA Pin	DDR SDRAM
DDR_E1_DATA7	U55.G17	U49.13
DDR_E1_DATA8	U55.K18	U49.54
DDR_E1_DATA9	U55.L18	U49.56
DDR_E1_ADD0	U55.C14	U49.29
DDR_E1_ADD1	U55.C15	U49.30
DDR_E1_ADD10	U55.E19	U49.28
DDR_E1_ADD11	U55.F19	U49.41
DDR_E1_ADD12	U55.D19	U49.42
DDR_E1_ADD13	U55.C19	U49.17
DDR_E1_ADD2	U55.L16	U49.31
DDR_E1_ADD3	U55.M16	U49.32
DDR_E1_ADD4	U55.J16	U49.35
DDR_E1_ADD5	U55.K16	U49.36
DDR_E1_ADD6	U55.H16	U49.37
DDR_E1_ADD7	U55.G16	U49.38
DDR_E1_ADD8	U55.G19	U49.39
DDR_E1_ADD9	U55.H19	U49.40
DDR_E1_LDM	U55.D16	U49.20
DDR_E1_LDQS	U55.D17	U49.16
DDR_E1_UDM	U55.C18	U49.47

Signal Name	FPGA Pin	DDR SDRAM
DDR_E1_UDQS	U55.M19	U49.51
DDR_E1_BA0	U55.C20	U49.26
DDR_E1_BA1	U55.D20	U49.27
DDR_E1_CASN	U55.K20	U49.22
DDR_E1_CKE	U55.M21	U49.44
DDR_E1_CSN	U55.J20	U49.24
DDR_E1_RASN	U55.L20	U49.23
DDR_E1_WEN	U55.H20	U49.21
DDR_E2_DATA0	U55.H10	U58.2
DDR_E2_DATA1	U55.J10	U58.4
DDR_E2_DATA10	U55.F14	U58.57
DDR_E2_DATA11	U55.G14	U58.59
DDR_E2_DATA12	U55.D14	U58.60
DDR_E2_DATA13	U55.E14	U58.62
DDR_E2_DATA14	U55.L15	U58.63
DDR_E2_DATA15	U55.K15	U58.65
DDR_E2_DATA2	U55.F10	U58.5
DDR_E2_DATA3	U55.G10	U58.7
DDR_E2_DATA4	U55.E10	U58.8
DDR_E2_DATA5	U55.D10	U58.10

Signal Name	FPGA Pin	DDR SDRAM
DDR_E2_DATA6	U55.C10	U58.11
DDR_E2_DATA7	U55.C11	U58.13
DDR_E2_DATA8	U55.K14	U58.54
DDR_E2_DATA9	U55.L14	U58.56
DDR_E2_ADD0	U55.G12	U58.29
DDR_E2_ADD1	U55.F12	U58.30
DDR_E2_ADD10	U55.F13	U58.28
DDR_E2_ADD11	U55.D13	U58.41
DDR_E2_ADD12	U55.C13	U58.42
DDR_E2_ADD13	U55.M15	U58.17
DDR_E2_ADD2	U55.D12	U58.31
DDR_E2_ADD3	U55.L13	U58.32
DDR_E2_ADD4	U55.M13	U58.35
DDR_E2_ADD5	U55.J13	U58.36
DDR_E2_ADD6	U55.K13	U58.37
DDR_E2_ADD7	U55.G13	U58.38
DDR_E2_ADD8	U55.H13	U58.39
DDR_E2_ADD9	U55.E13	U58.40
DDR_E2_LDM	U55.H11	U58.20
DDR_E2_LDQS	U55.F11	U58.16

Signal Name	FPGA Pin	DDR SDRAM
DDR_E2_UDM	U55.H15	U58.47
DDR_E2_UDQS	U55.F15	U58.51
DDR_E2_BA0	U55.F9	U58.26
DDR_E2_BA1	U55.E9	U58.27
DDR_E2_CASN	U55.H12	U58.22
DDR_E2_CKE	U55.K11	U58.44
DDR_E2_CSN	U55.K12	U58.24
DDR_E2_RASN	U55.J12	U58.23
DDR_E2_WEN	U55.L12	U58.21
DDR_F1_DATA0	U54.AN20	U47.2
DDR_F1_DATA1	U54.AM20	U47.4
DDR_F1_DATA10	U54.AV17	U47.57
DDR_F1_DATA11	U54.AV18	U47.59
DDR_F1_DATA12	U54.AN18	U47.60
DDR_F1_DATA13	U54.AM18	U47.62
DDR_F1_DATA14	U54.AU17	U47.63
DDR_F1_DATA15	U54.AT17	U47.65
DDR_F1_DATA2	U54.AP20	U47.5
DDR_F1_DATA3	U54.AR20	U47.7
DDR_F1_DATA4	U54.AV19	U47.8

Signal Name	FPGA Pin	DDR SDRAM
DDR_F1_DATA5	U54.AU19	U47.10
DDR_F1_DATA6	U54.AW19	U47.11
DDR_F1_DATA7	U54.AY19	U47.13
DDR_F1_DATA8	U54.AT18	U47.54
DDR_F1_DATA9	U54.AR18	U47.56
DDR_F1_ADD0	U54.AT19	U47.29
DDR_F1_ADD1	U54.AR19	U47.30
DDR_F1_ADD10	U54.AM17	U47.28
DDR_F1_ADD11	U54.AN17	U47.41
DDR_F1_ADD12	U54.AP16	U47.42
DDR_F1_ADD13	U54.AN16	U47.17
DDR_F1_ADD2	U54.AM19	U47.31
DDR_F1_ADD3	U54.AL19	U47.32
DDR_F1_ADD4	U54.AP19	U47.35
DDR_F1_ADD5	U54.AN19	U47.36
DDR_F1_ADD6	U54.AR17	U47.37
DDR_F1_ADD7	U54.AP17	U47.38
DDR_F1_ADD8	U54.AL18	U47.39
DDR_F1_ADD9	U54.AL17	U47.40
DDR_F1_BA0	U54.AW20	U47.26

Signal Name	FPGA Pin	DDR SDRAM
DDR_F1_BA1	U54.AY20	U47.27
DDR_F1_CASN	U54.AT16	U47.22
DDR_F1_CKE	U54.AW16	U47.44
DDR_F1_CSN	U54.AV16	U47.24
DDR_F1_LDM	U54.AL21	U47.20
DDR_F1_LDQS	U54.AV20	U47.16
DDR_F1_RASN	U54.AR16	U47.23
DDR_F1_UDM	U54.AW17	U47.47
DDR_F1_UDQS	U54.AY18	U47.51
DDR_F1_WEN	U54.AL15	U47.21
DDR_F2_DATA0	U54.AW14	U57.2
DDR_F2_DATA1	U54.AV14	U57.4
DDR_F2_DATA10	U54.AP13	U57.57
DDR_F2_DATA11	U54.AN13	U57.59
DDR_F2_DATA12	U54.AR12	U57.60
DDR_F2_DATA13	U54.AP12	U57.62
DDR_F2_DATA14	U54.AT12	U57.63
DDR_F2_DATA15	U54.AU12	U57.65
DDR_F2_DATA2	U54.AM15	U57.5
DDR_F2_DATA3	U54.AN15	U57.7

Signal Name	FPGA Pin	DDR SDRAM
DDR_F2_DATA4	U54.AU14	U57.8
DDR_F2_DATA5	U54.AT14	U57.10
DDR_F2_DATA6	U54.AN14	U57.11
DDR_F2_DATA7	U54.AM14	U57.13
DDR_F2_DATA8	U54.AM13	U57.54
DDR_F2_DATA9	U54.AL13	U57.56
DDR_F2_ADD0	U54.AV15	U57.29
DDR_F2_ADD1	U54.AU15	U57.30
DDR_F2_ADD10	U54.AV11	U57.28
DDR_F2_ADD11	U54.AU11	U57.41
DDR_F2_ADD12	U54.AY10	U57.42
DDR_F2_ADD13	U54.AY11	U57.17
DDR_F2_ADD2	U54.AY14	U57.31
DDR_F2_ADD3	U54.AY15	U57.32
DDR_F2_ADD4	U54.AV13	U57.35
DDR_F2_ADD5	U54.AU13	U57.36
DDR_F2_ADD6	U54.AW13	U57.37
DDR_F2_ADD7	U54.AY13	U57.38
DDR_F2_ADD8	U54.AN12	U57.39
DDR_F2_ADD9	U54.AM12	U57.40

Signal Name	FPGA Pin	DDR SDRAM
DDR_F2_LDM	U54.AR14	U57.20
DDR_F2_LDQS	U54.AR15	U57.16
DDR_F2_UDM	U54.AW12	U57.47
DDR_F2_UDQS	U54.AT13	U57.51
DDR_F2_BA0	U54.AL16	U57.26
DDR_F2_BA1	U54.AM16	U57.27
DDR_F2_CASN	U54.AV10	U57.22
DDR_F2_CKE	U54.AN11	U57.44
DDR_F2_CSN	U54.AR11	U57.24
DDR_F2_RASN	U54.AW10	U57.23
DDR_F2_WEN	U54.AU10	U57.21

7 Rocket IO Transceivers

The multigigabit transceivers (MGTs) can transmit data at speeds from 622 Mb/s up to 3.125 Gb/s (determined by the speed grade of the part, please refer to the Xilinx datasheet). MGTs are capable of various high-speed serial standards such as Gigabit Ethernet, FiberChannel, InfiniBand, and XAUI. In addition, the channel-bonding feature aggregates multiple channels, allowing for even higher data transfer rates. For additional information on RocketIO transceivers, see the *RocketIO Transceiver User Guide* at: <http://www.xilinx.com/publications/products/v2pro/userguide/ug024.pdf>

The DN6000K10PCI board has 10 RocketIO transceivers available on the top side of the FPGA and 10 on the bottom side. These 20 transceivers are connected in various configurations depending on the FPGA position on the board; refer to the block diagram for more information. FPGA A/B/E/F has access to two SMB interfaces, while the rest of the RocketIO interfaces are used for chip-to-chip communication. Refer to the RocketIO Block Diagram in [Figure 42](#).

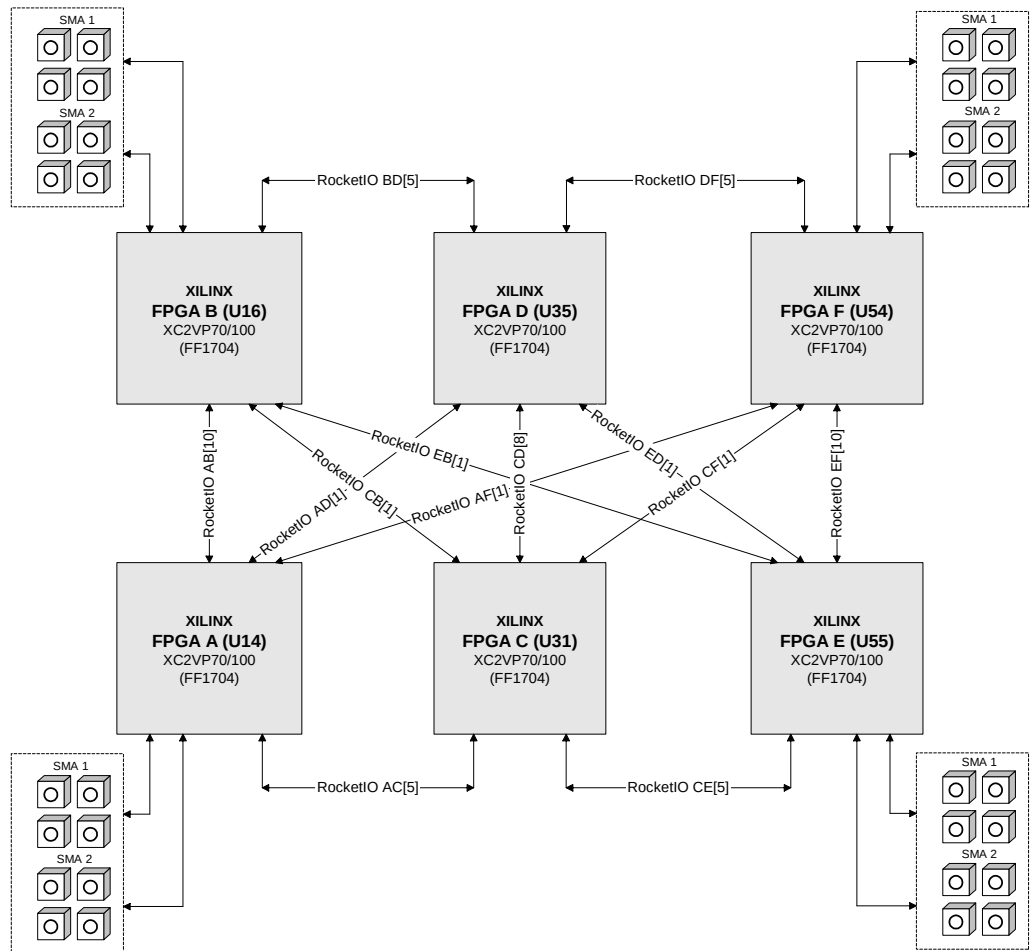


Figure 42 - RocketIO Block Diagram

7.1 SMB Connectors

The SMB connectors allow for direct connection the FPGA MGT interfaces.

7.1.1 FPGA to SMB Connector

The DN6000K10PCI board provides two discrete MGT channels for FPGA A/B/E/F. The connection between the FPGA and the SMA connectors is fairly simple, involving only one wire per connector, as well as a few capacitors and resistors to AC-couple the signals. These connections are also shown in [Table 19](#).

Table 19 - Connections between FPGA and SMA Connectors

Signal Name	FPGA Pin	Connector
FPGAA_SMB1_RXN	U14.A38	J12

Signal Name	FPGA Pin	Connector
FPGAA_SMB1_RXP	U14.A39	J9
FPGAA_SMB1_TXN	U14.A41	J8
FPGAA_SMB1_TXP	U14.A40	J11
FPGAA_SMB2_RXN	U14.A34	J24
FPGAA_SMB2_RXP	U14.A35	J18
FPGAA_SMB2_TXN	U14.A37	J17
FPGAA_SMB2_TXP	U14.A36	J23
FPGAB_SMB1_RXN	U16.BB34	J13
FPGAB_SMB1_RXP	U16.BB35	J14
FPGAB_SMB1_TXN	U16.BB37	J20
FPGAB_SMB1_TXP	U16.BB36	J19
FPGAB_SMB2_RXN	U16.BB38	J15
FPGAB_SMB2_RXP	U16.BB39	J16
FPGAB_SMB2_TXN	U16.BB41	J22
FPGAB_SMB2_TXP	U16.BB40	J21
FPGA_E_SMB1_RXN	U55.A6	J47
FPGA_E_SMB1_RXP	U55.A7	J48
FPGA_E_SMB1_TXN	U55.A9	J40
FPGA_E_SMB1_TXP	U55.A8	J39
FPGA_E_SMB2_RXN	U55.A2	J45

Signal Name	FPGA Pin	Connector
FPGA_E_SMB2_RXP	U55.A3	J46
FPGA_E_SMB2_TXN	U55.A5	J38
FPGA_E_SMB2_TXP	U55.A4	J37
FPGA_F_SMB1_RXN	U54.BB2	J44
FPGA_F_SMB1_RXP	U54.BB3	J43
FPGA_F_SMB1_TXN	U54.BB5	J35
FPGA_F_SMB1_TXP	U54.BB4	J36
FPGA_F_SMB2_RXN	U54.BB6	J42
FPGA_F_SMB2_RXP	U54.BB7	J41
FPGA_F_SMB2_TXN	U54.BB9	J33
FPGA_F_SMB2_TXP	U54.BB8	J34

Please note the RocketIO Transceiver performance in [Table 20](#):

Table 20 - RocketIO Performance

Item	Speed Grade			Units
	-7	-6	-5	
RocketIO Transceiver (FF)	3.125	3.125	2.0	Gb/s
PowerPC Processor Block	400	350	300	MHz

8 CPU Debug and CPU Trace

The DN6000K10PCI board includes two CPU debugging interfaces for FPGA A/B, the CPU Debug (vertical headers, i.e., JP1 and JP2) and the Combined CPU Trace and Debug, (vertical micro connector, i.e., J10 and J5). These connectors can be used in conjunction with third party tools, or in some cases the Xilinx Parallel Cable IV, to debug software as it runs on the processor.

The PowerPC™ 405 CPU core includes dedicated debug resources that support a variety of debug modes for debugging during hardware and software development. These debug resources include:

- Internal debug mode for use by ROM monitors and software debuggers
- External debug mode for use by JTAG debuggers
- Debug wait mode, which allows the servicing of interrupts while the processor appears to be stopped
- Real-time trace mode, which supports event triggering for real-time tracing

Debug modes and events are controlled using debug registers in the processor. The debug registers are accessed either through software running on the processor or through the JTAG port. The debug modes, events, controls, and interfaces provide a powerful combination of debug resources for hardware and software development tools.

The JTAG port interface supports the attachment of external debug tools, such as the ChipScope™ Integrated Logic Analyzer, a powerful tool providing logic analyzer capabilities for signals inside an FPGA, without the need for expensive external instrumentation. Using the JTAG test access port, a debug tool can single-step the processor and examine the internal processor state to facilitate software debugging. This capability complies with the IEEE 1149.1 specification for vendor-specific extensions and is, therefore, compatible with standard JTAG hardware for boundary-scan system testing.

8.1 CPU Debug

External-debug mode can be used to alter normal program execution. It provides the ability to debug system hardware as well as software. The mode supports multiple functions: starting and stopping the processor, single-stepping instruction execution, setting breakpoints, as well as monitoring processor status. Access to processor resources is provided through the CPU Debug port.

The PPC405 JTAG (Joint Test Action Group) Debug port complies with IEEE standard 1149.1-1990, IEEE Standard Test Access Port and Boundary Scan Architecture. This standard describes a method for accessing internal chip resources using a four-signal or five-signal interface. The PPC405 JTAG Debug port supports scan-based board testing and is further enhanced to support the attachment of debug tools. These enhancements comply with the IEEE 1149.1 specifications for vendor-specific extensions and are compatible with standard JTAG hardware for boundary-scan system testing.

The PPC405 JTAG debug port supports the four required JTAG signals: TCK, TMS, TDI, and TDO. It also implements the optional TRST signal. The frequency of the JTAG clock signal can range from 0 MHz (DC) to one-half of the processor clock frequency. The JTAG debug port logic is reset at the same time the system is reset, using TRST. When TRST is asserted, the JTAG TAP controller returns to the test-logic reset state.

Refer to the *PPC405 Processor Block Manual* for more information on the JTAG debug-port signals. Information on JTAG is found in the IEEE standard 1149.1-1990.

8.1.1 CPU Debug Connectors

Figure 43 shows JP2, the vertical header used to debug the operation of software in the PPC of FPGA A (there is another connector on FPGA B). This is done using debug tools such as Parallel Cable IV or third party tools. This connector cannot be used when the Mictor connector is in use.

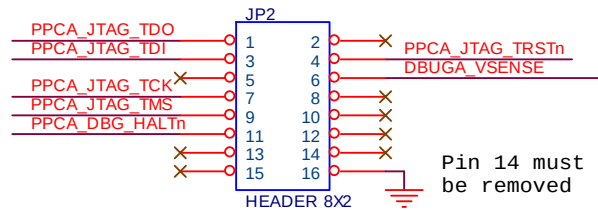


Figure 43 - CPU Debug Connector

8.1.2 CPU Debug Connection to FPGA's

The connection between the PPC debug connectors and the FPGA's are shown in Table 21. These signals are attached to the PowerPC™ 405 JTAG debug resources using normal FPGA routing resources. The JTAG debug resources are not hard-wired to particular pins, and are available for attachment in the FPGA fabric, making it is possible to route these signals to whichever FPGA pins the user would prefer to use.

Table 21 - CPU Debug connection to FPGA

Signal Name	FPGA Pin	Connector
PPCA_DBG_HALTn	JP2.11	U14.AC32
PPCA_JTAG_TCK	JP2.7	U14.AC33
PPCA_JTAG_TDI	JP2.3	U14.AC36
PPCA_JTAG_TDO	JP2.1	U14.AC37

Signal Name	FPGA Pin	Connector
PPCA_JTAG_TMS	JP2.9	U14.AC34
PPCA_JTAG_TRSTN	JP2.4	GND
PPCB_DBG_HALTN	JP1.11	U16.AJ42
PPCB_JTAG_TCK	JP1.7	U16.AJ34
PPCB_JTAG_TDI	JP1.3	U16.AJ38
PPCB_JTAG_TDO	JP1.1	U16.AJ41
PPCB_JTAG_TMS	JP1.9	U16.AJ37
PPCB_JTAG_TRSTN	JP1.4	GND

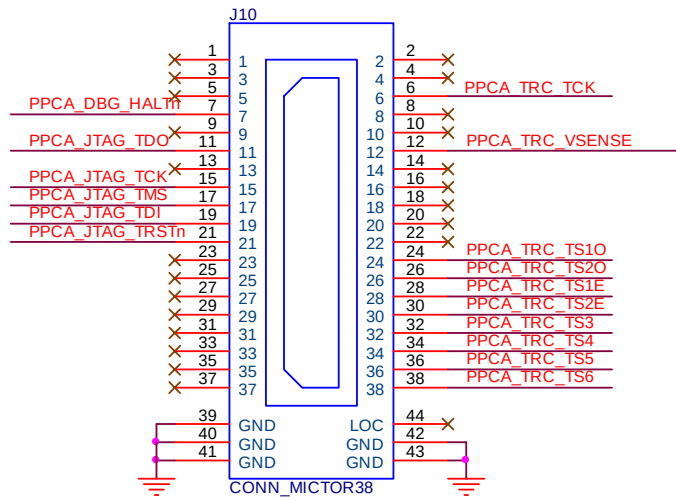
8.2 CPU Trace

The CPU Trace port accesses the real-time, trace-debug capabilities built into the PowerPC™ 405 CPU core. Real-time trace-debug mode supports real-time tracing of the instruction stream executed by the processor. In this mode, debug events are used to cause external trigger events. An external trace tool uses the trigger events to control the collection of trace information. The broadcast of trace information occurs independently of external trigger events (trace information is always supplied by the processor).

Real-time trace-debug does not affect processor performance. Real-time trace-debug mode is always enabled. However, the trigger events occur only when both internal-debug mode and external debug mode are disabled. Most trigger events are blocked when either of those two debug modes is enabled. Information on the trace-debug capabilities, how trace-debug works, and how to connect an external trace tool is available in the *RISCWatch Debugger User's Guide*.

8.2.1 CPU Trace Connectors

[Figure 44](#) shows J10, the vertical header used to trace the operation of software in the PPC of FPGA A (there is another connector on FPGA B). Agilent/WinDriver has defined a Trace Port Analyzer (TPA) port for the PowerPC 4xx line of CPU cores that combines the CPU Trace and the CPU Debug interfaces onto a single 38-pin Mictor connector. This provides for high-speed, controlled-impedance signaling.



Note: All these signals must be matched length $\pm 1/50$ mils.

Figure 44 - Combined Trace/Debug Connector Pinout

8.2.2 Combined CPU Trace/Debug Connection to FPGA's

The connection between the Combined CPU Trace and Debug Port connectors is shown in Table 22. The connections to the FPGA are shared with the CPU Trace and CPU Debug interfaces discussed in previous sections.

Table 22 - Combined CPU Trace/Debug connection to FPGA

Signal Name	FPGA Pin	Connector
PPCA_DBG_HALTn	J10.7	U14.AC32
PPCA_JTAG_TCK	J10.15	U14.AC33
PPCA_JTAG_TDI	J10.19	U14.AC36
PPCA_JTAG_TDO	J10.11	U14.AC37
PPCA_JTAG_TMS	J10.17	U14.AC34
PPCA_JTAG_TRSTn	J10.21	GND
PPCA_TRC_TCK	J10.6	U14.AC31
PPCA_TRC_TS1E	J10.28	U14.AB33

Signal Name	FPGA Pin	Connector
PPCA_TRC_TS1O	J10.24	U14.AC39
PPCA_TRC_TS2E	J10.30	U14.AB34
PPCA_TRC_TS2O	J10.26	U14.AC40
PPCA_TRC_TS3	J10.32	U14.AB36
PPCA_TRC_TS4	J10.34	U14.AB37
PPCA_TRC_TS5	J10.36	U14.AB39
PPCA_TRC_TS6	J10.38	U14.AB40
PPCA_TRC_VSENSE	J10.12	GND
PPCB_DBG_HALTN	J5.7	U16.AJ42
PPCB_JTAG_TCK	J5.15	U16.AJ34
PPCB_JTAG_TDI	J5.19	U16.AJ38
PPCB_JTAG_TDO	J5.11	U16.AJ41
PPCB_JTAG_TMS	J5.17	U16.AJ37
PPCB_JTAG_TRSTN	J5.21	GND
PPCB_TRC_TCK	J5.6	U16.AJ33
PPCB_TRC_TS1E	J5.28	U16.AK31
PPCB_TRC_TS1O	J5.24	U16.AK39
PPCB_TRC_TS2E	J5.30	U16.AK32
PPCB_TRC_TS2O	J5.26	U16.AK40
PPCB_TRC_TS3	J5.32	U16.AK41

Signal Name	FPGA Pin	Connector
PPCB_TRC_TS4	J5.34	U16.AK42
PPCB_TRC_TS5	J5.36	U16.AJ35
PPCB_TRC_TS6	J5.38	U16.AJ36
PPCB_TRC_VSENSE	J5.12	GND

9 GPIO LED's

9.1 Status Indicators

The DN6000K10PCI uses DS1 and DS2 to visually indicate the status of the board. DS1 is controller by the MCU (U65) and the Configuration FPGA (U2) controls DS2.

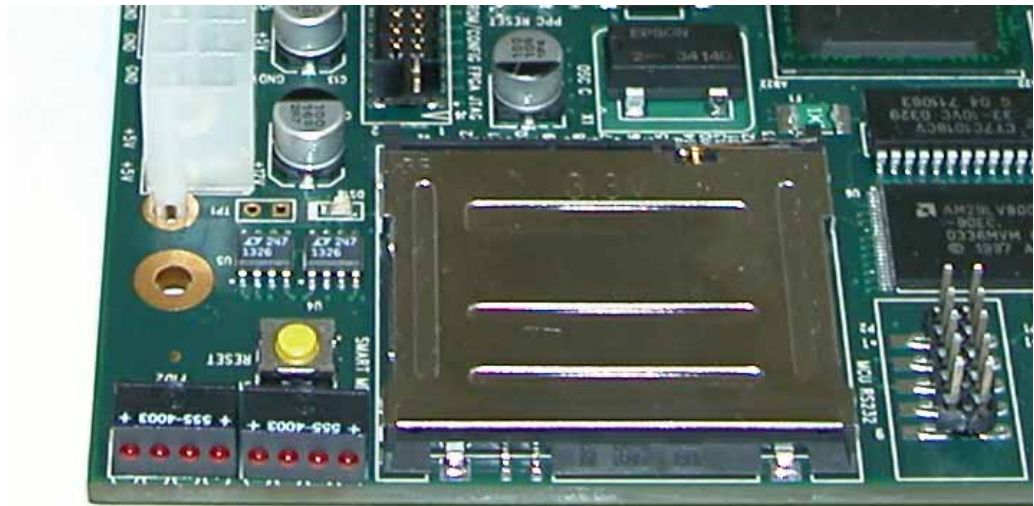


Table 23 lists the function of the CPLD LED's. The LED's is number from left to right CPLD_LED0n to CPLD_LED3n.

Table 23 - CPLD LED's

Signal Name	Device	LED	Description
CFPGA_LEDn0	U2.L1	DS1.1	Blinks when configuring over USB
CFPGA_LEDn1	U2.L5	DS1.2	Blinks when reading data from the SmartMedia card

Signal Name	Device	LED	Description
CFPGA_LEDn2	U2.L4	DS1.3	Blinks when MCU is reading/writing data to/from the FPGA's
CFPGA_LEDn3	U2.L3	DS1.4	Blinks when the MCU is read/writing data to/from the FPGA's via the USB interface

The MCU_LED's are used to show which FPGA is currently being configured (either by SmartMedia or over USB), and also give the user overall configuration status.

Table 24 - MCU LED's

FPGA / Status	MCU_LED0n	MCU_LED1n	MCU_LED2n	MCU_LED3n
A	On	Off	Off	off
B	Off	On	Off	Off
C	On	On	Off	Off
D	Off	Off	On	Off
E	On	Off	On	Off
F	Off	On	On	Off
Successful Configuration	Off	Off	On	On
Error during Configuration / No FPGAs configured	Blink	Blink	Blink	Blink

9.2 FPGA A GPIO LED's

The DN6000K10PCI provides 10 GPIO LED's directly connected to FPGA A IO Bank 2 pins. Table 25 lists the FPGA GPIO LED's on the DN6000K10PCI and is available to the user. The signals are active LOW.

Table 25 – FPGA A GPIO LED's

Signal Name	FPGA A	LED
LED0	U14.P11	DS13
LED1	U14.P12	DS14
LED2	U14.R11	DS16

Signal Name	FPGA A	LED
LED3	U14.R12	DS15
LED4	U14.T11	DS18
LED5	U14.T12	DS17
LED6	U14.U12	DS19
LED7	U14.V11	DS22
LED8	U14.U11	DS20
LED9	U14.V12	DS21

10 PCI Interface

Peripheral Component Interconnect (PCI) Local Bus is a bus standard that is a mainstay of many different computer systems. PCI is a high-performance bus with multiplexed address and data lines. Defined for both 32-bit and 64-bit wide data buses, PCI is intended for use as an interconnect mechanism between highly integrated peripheral controller components, peripheral add-in boards, and processor/memory systems. The DN6000K10PCI can be hosted in a 32-bit or 64-bit PCI/PCI-X slot and includes two main components:

- FPGA A as the PCI bus Master
- PCI Edge Connector

Virtex-II Pro parts do not tolerate +5V signaling, so the DN6000K10PCI must be plugged into a +3.3V PCI slot (PCI-X, by definition, is +3.3V signaling). The PWB is keyed so that it is not possible to mistakenly plug the board into a +5V PCI slot. Do **NOT** grind out the key in the PCI host slot, and do **NOT** modify the DN6000K10PCI to get it to fit into the slot. If you need a +3.3V PCI slot, the DNPCIEXT-S3 Extender card can perform this function. Please refer to the Dini Group website. The extender also has the capability to slow the clock frequency of the PCI bus by a factor of two - function that is very useful when prototyping ASIC's.

Note: The PCI interface is not 5V tolerant. Do not modify the PCI edge connector to fit in the host PC.

The +3.3V power on the PCI connector is not used. Instead, you must supply power through the ATX power connector (J2). Please note you should turn on the motherboard power supply first and then the ATX power supply. To turn off the

power to the DN6000k10PCI you must turn off the ATX power supply first and then the motherboard power supply.

Note: The DN6000k10PCI requires power through the ATX power supply connector (J2) even when the board is plugged into PCI. The motherboard power supply must be turned on first and then the ATX power supply.

10.1 Connection to the FPGA

The FPGA connections to the PCI bus consist of 91 signals in Bank 0. A description of these signals can be found in the following sections.

10.1.1 PCI VCCO on the FPGA

A Linear Technology LTC1763 regulator (refer to Figure 45) is used to ensure electrical compatibility to PCI and to protect the Virtex-II Pro from over-voltage conditions. It is used for the VCCO of the banks connected to the PCI interface. For more information, see XAPP653: Virtex-II Pro PCI Reference Design at:

<http://www.xilinx.com/xapp/xapp653.pdf>.

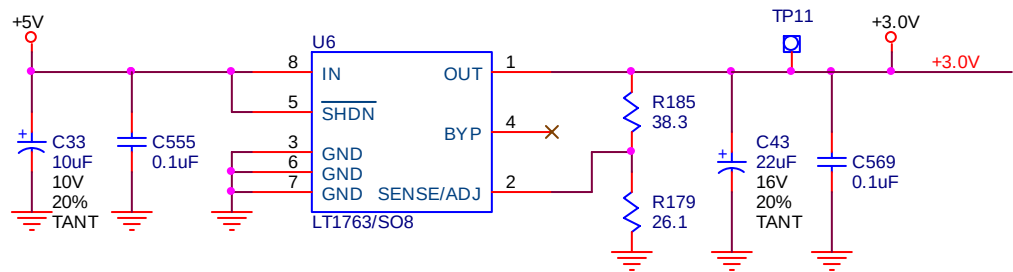


Figure 45 - VirtexII Pro PCI VCCO Regulator

10.1.2 PCI Edge Connector

Figure 46 shows P4, the PCI 3.3V 64-Bit edge connector used to interface with the host PC.

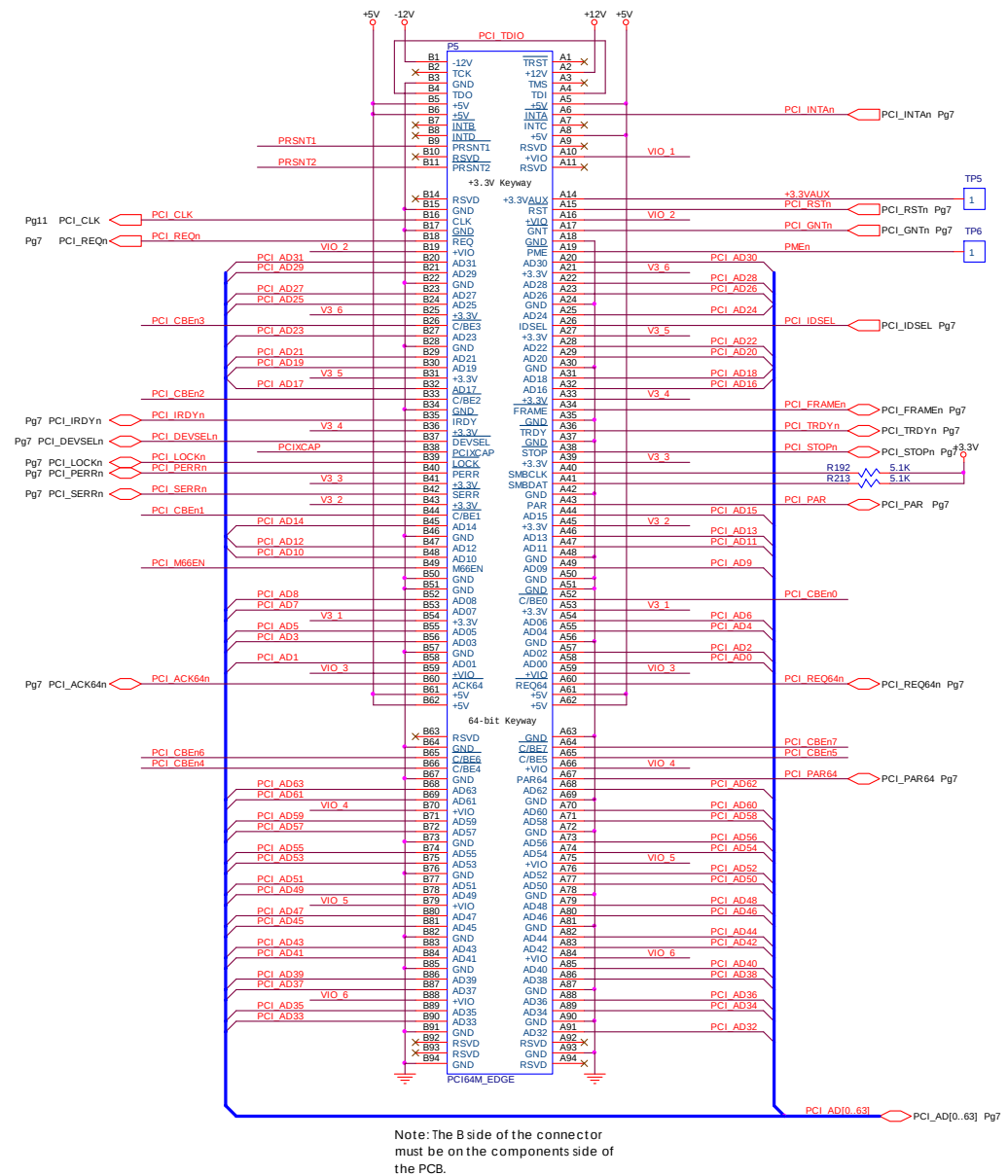


Figure 46 - PCI Edge Connector

10.1.3 Connection between the PCI connector and the FPGA

Table 26 shows the connection between the PCI Edge Connector and the FPGA.

Table 26 - PCI to FPGA Connections

Signal Name	Connector	FPGA Pin
PCI_AD0	P5.A58	U14.J27

Signal Name	Connector	FPGA Pin
PCI_AD1	P5.B58	U14.K27
PCI_AD2	P5.A57	U14.H27
PCI_AD3	P5.B56	U14.G27
PCI_AD4	P5.A55	U14.E27
PCI_AD5	P5.B55	U14.F27
PCI_AD6	P5.A54	U14.D27
PCI_AD7	P5.B53	U14.M28
PCI_AD8	P5.B52	U14.F28
PCI_AD9	P5.A49	U14.C28
PCI_AD10	P5.B48	U14.M29
PCI_AD11	P5.A47	U14.M30
PCI_AD12	P5.B47	U14.C29
PCI_AD13	P5.A46	U14.L30
PCI_AD14	P5.B45	U14.K30
PCI_AD15	P5.A44	U14.H30
PCI_AD16	P5.A32	U14.E31
PCI_AD17	P5.B32	U14.F31
PCI_AD18	P5.A31	U14.D31
PCI_AD19	P5.B30	U14.K32
PCI_AD20	P5.A29	U14.H32

Signal Name	Connector	FPGA Pin
PCI_AD21	P5.B29	U14.J32
PCI_AD22	P5.A28	U14.F32
PCI_AD23	P5.B27	U14.E32
PCI_AD24	P5.A25	U14.H33
PCI_AD25	P5.B24	U14.G33
PCI_AD26	P5.A23	U14.E33
PCI_AD27	P5.B23	U14.F33
PCI_AD28	P5.A22	U14.D33
PCI_AD29	P5.B21	U14.C33
PCI_AD30	P5.A20	U14.G34
PCI_AD31	P5.B20	U14.H34
PCI_AD32	P5.A91	U14.M22
PCI_AD33	P5.B90	U14.M23
PCI_AD34	P5.A89	U14.K23
PCI_AD35	P5.B89	U14.L23
PCI_AD36	P5.A88	U14.J23
PCI_AD37	P5.B87	U14.H23
PCI_AD38	P5.A86	U14.E23
PCI_AD39	P5.B86	U14.F23
PCI_AD40	P5.A85	U14.D23

Signal Name	Connector	FPGA Pin
PCI_AD41	P5.B84	U14.C23
PCI_AD42	P5.A83	U14.L24
PCI_AD43	P5.B83	U14.M24
PCI_AD44	P5.A82	U14.K24
PCI_AD45	P5.B81	U14.J24
PCI_AD46	P5.A80	U14.G24
PCI_AD47	P5.B80	U14.H24
PCI_AD48	P5.A79	U14.F24
PCI_AD49	P5.B78	U14.E24
PCI_AD50	P5.A77	U14.C24
PCI_AD51	P5.B77	U14.D24
PCI_AD52	P5.A76	U14.M25
PCI_AD53	P5.B75	U14.L25
PCI_AD54	P5.A74	U14.H25
PCI_AD55	P5.B74	U14.K25
PCI_AD56	P5.A73	U14.G25
PCI_AD57	P5.B72	U14.E25
PCI_AD58	P5.A71	U14.M26
PCI_AD59	P5.B71	U14.C25
PCI_AD60	P5.A70	U14.L26

Signal Name	Connector	FPGA Pin
PCI_AD61	P5.B69	U14.K26
PCI_AD62	P5.A68	U14.H26
PCI_AD63	P5.B68	U14.J26
PCI_CBEN0	P5.A52	U14.E28
PCI_CBEN1	P5.B44	U14.J30
PCI_CBEN2	P5.B33	U14.G31
PCI_CBEN3	P5.B26	U14.C32
PCI_CBEN4	P5.B66	U14.F26
PCI_CBEN5	P5.A65	U14.D26
PCI_CBEN6	P5.B65	U14.E26
PCI_CBEN7	P5.A64	U14.C26
PCI_CLK	P5.B16	U14.G22
PCI_DEVSEL _n	P5.B37	U14.L31
PCI_FRAME _n	P5.A34	U14.H31
PCI_GNT _n	P5.A17	U14.E34
PCI_IDSEL	P5.A26	U14.J33
PCI_INTA _n	P5.A6	U14.C34
PCI_IRDY _n	P5.B35	U14.J31
PCI_LOCK _n	P5.B39	U14.D30
PCI_PAR	P5.A43	U14.G30

Signal Name	Connector	FPGA Pin
PCI_PAR64	P5.A67	U14.G26
PCI_PERRn	P5.B40	U14.E30
PCI_REQ64n	P5.A60	U14.L27
PCI_ACK64n	P5.B60	U14.F34
PCI_REQn	P5.B18	U14.F34
PCI_RSTn	P5.A15	U14.D34
PCI_SERRn	P5.B42	U14.F30
PCI_STOPn	P5.A38	U14.C30
PCI_TRDYn	P5.A36	U14.K31

10.2 PCI/PCI-X Hardware Setup

The following section describes the PCI/PCI-X hardware setup. More information is available from the PCI/PCI-X Specifications, available from PCI-SIG: <http://www.pcisig.com/home>.

10.2.1 Present Signals

The Present signals indicate to the system board whether an add-in card is physically present in the slot and, if one is present, the total power requirements of the add-in card (refer to Table 27).

Table 27 - Present Signal Definition

PRSNT1#	PRSNT2#	Add-in Card Configuration
Open	Open	No add-in card present
Ground	Open	Add-in card present, 25 W maximum
Open	Ground	Add-in card present, 15 W maximum
Ground	Ground	Add-in card present, 7.5 W maximum

The DN6000K10PCI is factory configured for 25W power setting (JP7.1-2 and JP7.3-4, jumpers installed).

10.2.2 M66EN and PCIXCAP Encoding

The 66MHZ_ENABLE pin indicates to a device whether the bus segment is operating at 66 or 33 MHz. Add-in cards indicate whether they support PCI-X, and if so which frequency, by the way they connect one pin called PCIXCAP (refer to Figure 47).

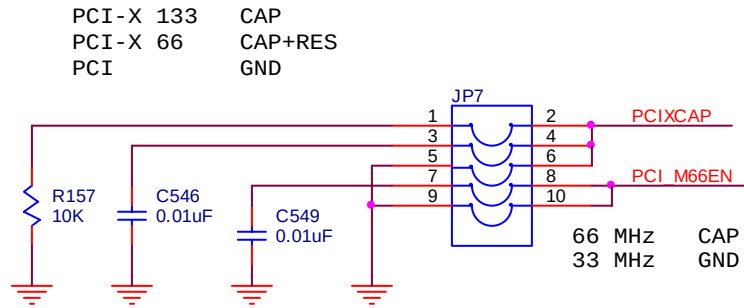


Figure 47 - M66EN and PCIXCAP Jumper

If the card's maximum frequency is 133 MHz, it leaves this pin unconnected (except for a decoupling capacitor). If the card's maximum frequency is 66 MHz, it connects PCIXCAP to ground through a resistor (and decoupling capacitor). Conventional cards connect this pin to ground. An add-in card indicates its capability with one of the combinations of the M66EN and PCIXCAP pins listed in Table 28.

Table 28 - M66EN and PCIXCAP Encoding

M66EN	PCIXCAP	Conventional Device Frequency Capability	PCI-X Device Frequency Capability
Ground	Ground	33 MHz	Not capable
Not connected	Ground	66 MHz	Not capable
Ground	Pull-down	33 MHz	PCI-X 66 MHz
Not connected	Pull-down	66 MHz	PCI-X 66 MHz
Ground	Not connected	33 MHz	PCI-X 133 MHz
Not connected	Not connected	66 MHz	PCI-X 133 MHz

The DN6000K10PCI is factory configured to operate in PCI mode at 33MHz (JP2.5-6 and JP2.9-10, jumpers installed).

10.2.3 Further Information on PCI/PCI-X Signals

The following signals have pull-up resistors (1M) on the DN6000K10PCI. This is technically a violation of the PCI specification, but there are systems that have these signals floating:

- PCI_LOCK_n
- PCI_REQ64_n
- PCI_ACK64_n

Note: The function of LOCK_n pin was deleted in version 2.3 of the PCI Specification. The PCI JTAG signals TDI, TDO, TCK, TRST_n, are not used. TDI and TDO are connected together per the PCI Specification to maintain JTAG chain integrity on the motherboard. The signals TMS, TCK, and TRST_n are left unconnected.

The following signals are not connected on the DN6000K10PCI:

- +3.3VAUX
- INTB_n, INTC_n, INTD_n

11 Power System

The DN6000K10PCI supports a wide range of technologies, from legacy devices like serial ports, to DDR SDRAM and RocketIO multi-gigabit transceivers (MGTs). This wide range of technologies requires a wide range of power supplies. These are provided on the DN6000K10PCI using a combination of switching and linear power regulators.

11.1 Stand Alone Operation

An external ATX power supply is used to supply power to the DN6000K10PCI (refer to [Figure 48](#)). The external power supply connects to header P16, Molex type header P/N 39-29-9202.

The DN6000K10PCI has the following power supplies:

- +1.25V
- +1.5V
- +2.5V
- +3.3V

- +5V
- +12V

The +1.5V, +2.5V power supplies are generated from the +5V supply on the External ATX power supply, while +3.3V comes directly from the ATX power supply.

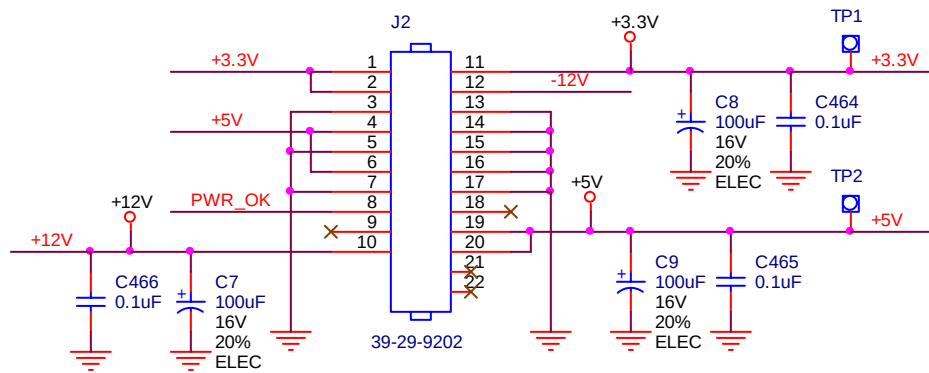


Figure 48 - ATX Power Supply

Any ATX type power supply is adequate. The Dini Group recommends a power supply rated for 250W. Note: The switching regulators in the Power Supply may require an external load to operate within specifications (the DN6000K10PCI may not meet the minimum load requirements). The Dini Group recommends attaching an old disk drive to one of the spare connectors.

11.1.1 External Power Connector

Figure 49 indicates the connections to the external power connector. This header is fully polarized to prevent reverse connection and is rated for 1500VAC at 6A per contact.



Note: Pin 14 is connected the GND,
PSU always "ON" configuration.

Figure 49 - External Power Connection

Note: Header J2 is not hot-plug able. Do not attach power while power supply is ON.

11.1.2 Power Monitors

Power supply monitor (U36) is used to monitor the +1.5V, +2.5V, +3.3V, and +5V supplies (for more information on these devices, please refer to the datasheet for the LT2900 from Linear Technology). The power supply monitor also provides a push-button reset input that is utilized to reset the various sub-circuits of the DN6000K10PCI. After power-up, SYS_RSTn remains asserted for approximately 10ms.

11.1.3 Power Indicators

There are six LED's on the DN6000K10PCI used to indicate the presence of the following voltage sources (refer to [Table 29](#)):

Table 29 – Voltage Indicators

Voltage Source	LED
PWR_OK	DS3
+2.5V	DS7
+3.3V	DS6
+5V	DS5
+12V	DS4
-12V	DS8

12 Test Header & Daughter Card Connections

12.1 Test Header

The DN6000K10PCI offers two 200-pin test headers (P6, P8) that allow the user connection to discrete FPGA pins, refer to [Figure 50](#), Test Header A is shown:

Note: Use of a Daughter card requires the FPGA fan to be removed, leaving the heatsink in place.

BOARD HARDWARE

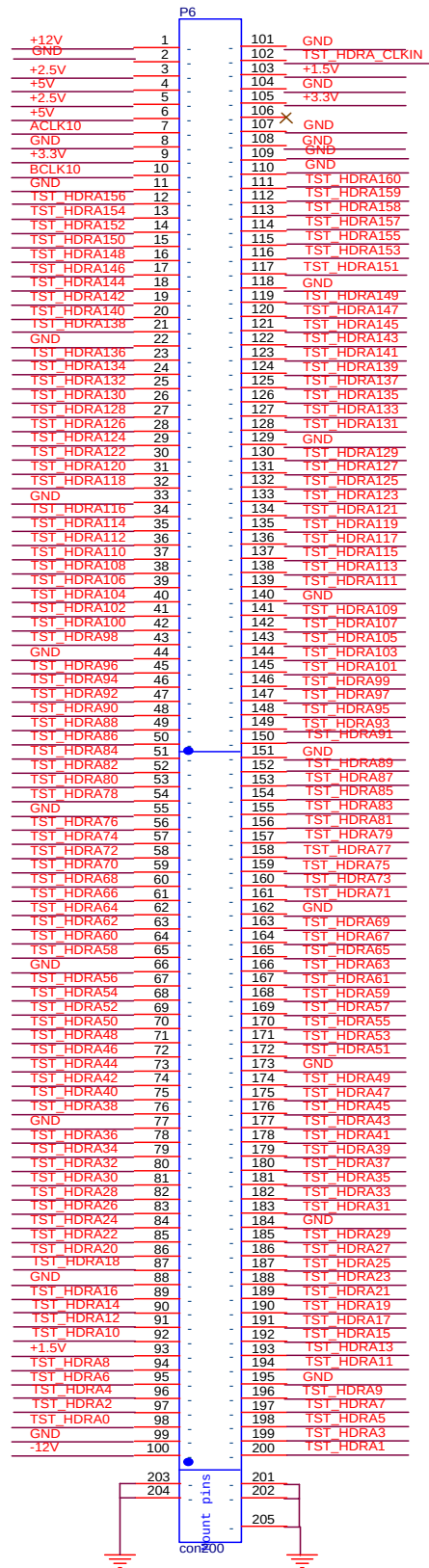


Figure 50 - Test Header

12.1.1 Test Header Connector

Micropax connector (200 pin) is used as a standard interface to all the Dini Group logic emulation boards. This connector has a specified current rating of 0.5 amps per contact. See datasheet for more information P/N 91294-003.

12.1.2 Test Header Pin Numbering

Figure 51 indicates the pin numbering scheme used on the test headers.

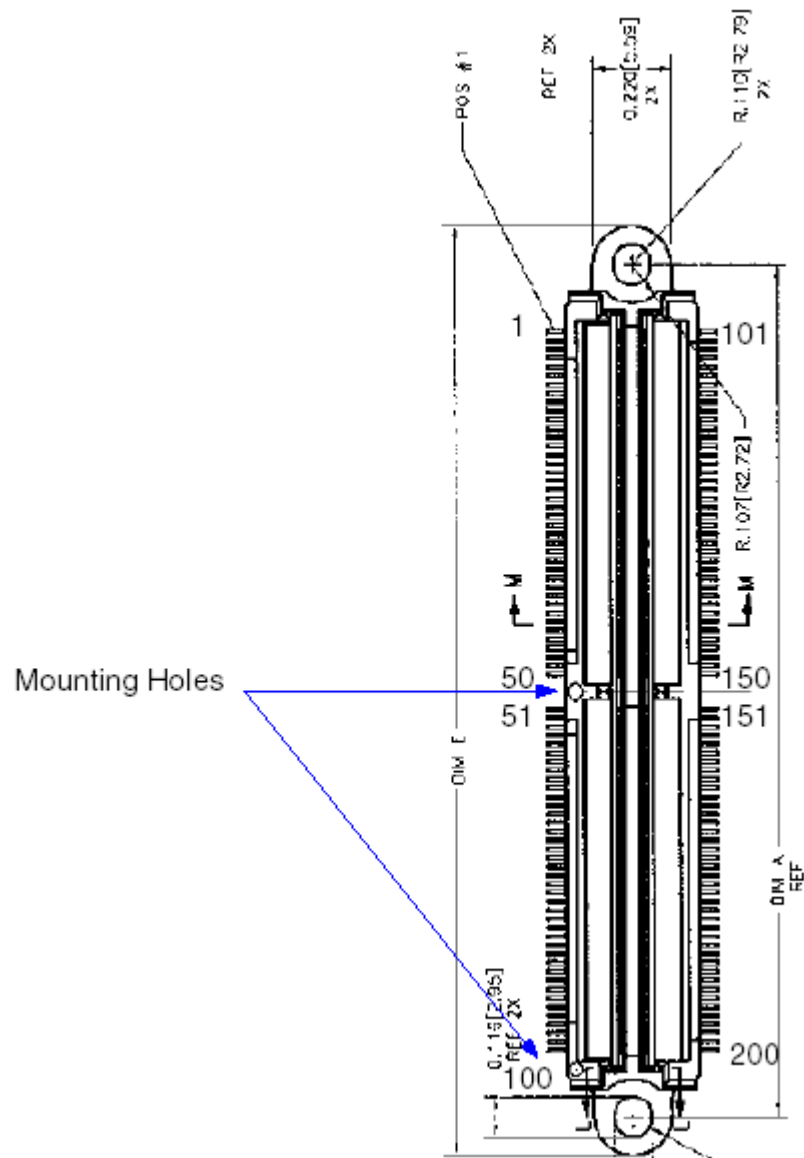


Figure 51 - Test Header Pin Numbering

12.2 DN3000K10SD Daughter Card

The Dini Group manufactures a daughter “DN3000K10SD” card that allows the user connection to the FPGA IO pins. The daughter card has the following features:

- Buffered I/O, Passive and Active Bus Drivers
- Unbuffered I/O
- Differential LVDS pairs (Note: Not available on DN6000K10PCI Logic Emulation board)
- Headers for Test Points

The daughter card contains headers that may be useful with certain types of oscilloscope probes, or when wiring pins to prototype areas. Figure 52 is a block diagram of the daughter card.

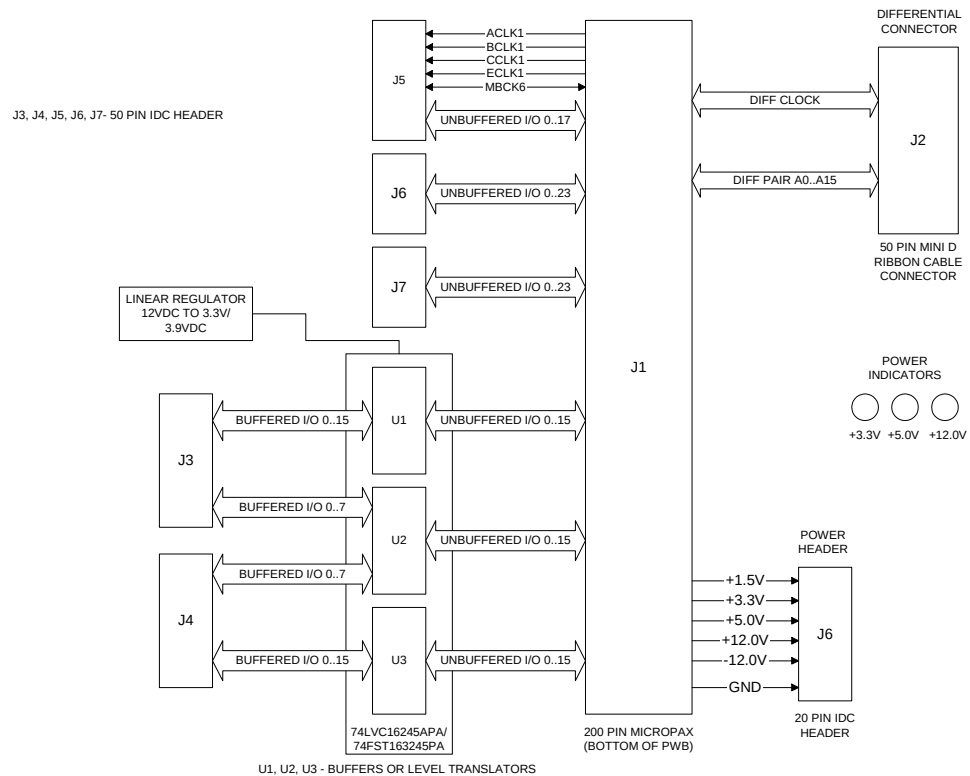


Figure 52 - DN3000K10SD Daughter Card Block Diagram

The DN3000K10SD Daughter Card provides 16-differential pairs, 48-buffered (passive/active) I/O, and 66-unbuffered I/O signals. The DN3000K10SD Daughter Card is pictured in Figure 53.

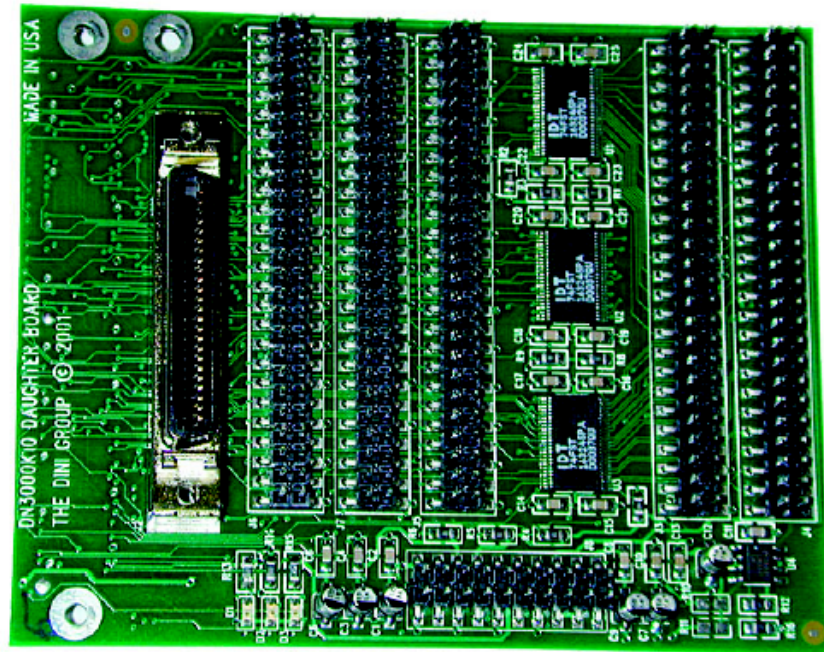
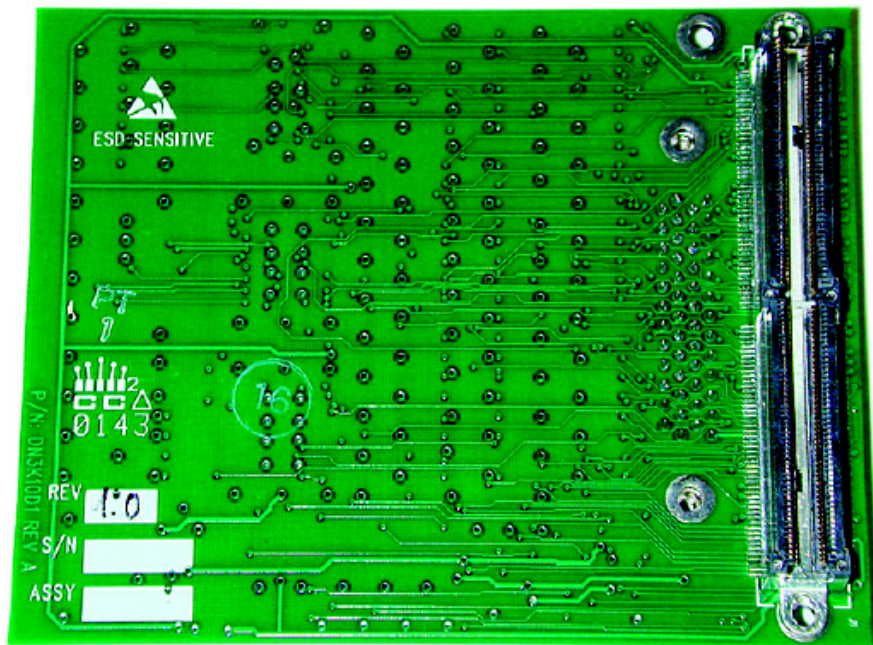
**Top****Bottom**

Figure 53 - DN3000K10S Daughter Card

Figure 54 show the assembly drawing of the DN3000K10SD Daughter Card. IDT74FST163245 devices (U1, U2, U3) are used as bus switches in the passive mode,

and the IDT74LVC16245A (U1, U2, U3) devices are used as bus transceivers in the active mode. The DN3000K10SD has separate enable/direction signals for each driver.

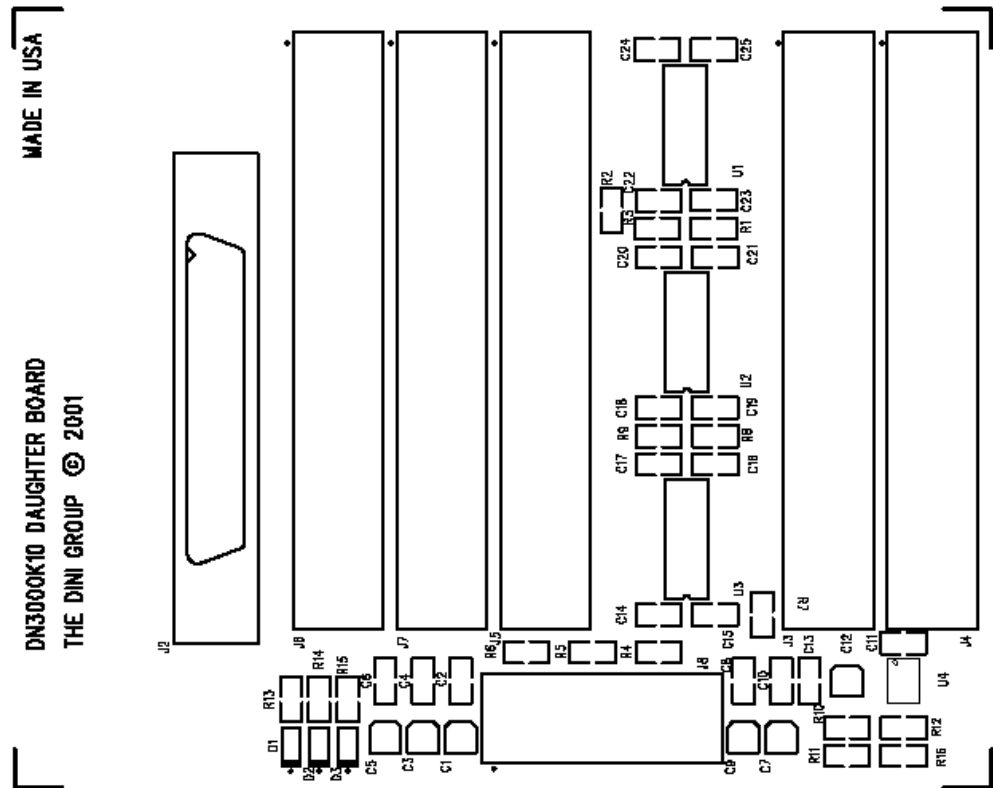


Figure 54 - Assembly drawing for the DN3000K10SD

NOTE: Signals P4NX7 and P4NX6 are also used for direction select and output enable on U2 and U3 respectively.

12.2.1 Daughter Card LED's

The LED's act as visual indicators, representing the presence of active power sources.

- D1 - LED indicating +3.3 V present
- D2 - LED indicating +5.0 V present
- D3 - LED indicating +12 V present

Under normal operating conditions, all LED's should be ON.

12.2.2 Power Supply

A linear power supply (U4) is present to provide level shift/translation functions when the board is populated with passive bus switches. Resistors R10 and R11 can be used to select alternate voltage sources, +5V or +3.3V, respectively. When used, U4 must be removed in order to prevent contention. The power supplies is rated as follows:

- +5 V power supply is rated for 1 A
- +3.3 V power supply is rated for 1 A
- +1.5 V power supply is rated for 1 A
- +12 V power supply is rated for 0.5 A
- -12 V power supply is rated for 0.5 A

NOTE: Never populate R10/R11 simultaneously, this will result in a shorting the +3.3V and +5V power supplies.

Header J8 allows external connection to the Power Sources (refer to [Table 30](#) for connection details).

Table 30 - External Power Connections

Pin	Function	Pin	Function
J8.1	GND	J8.11	GND
J8.2	+5V	J8.12	+1.5V
J8.3	GND	J8.13	GND
J8.4	+5V	J8.14	+12V
J8.5	GND	J8.15	GND
J8.6	+3.3V	J8.16	+12V
J8.7	GND	J8.17	GND
J8.8	+3.3V	J8.18	-12V
J8.9	GND	J8.19	GND
J8.10	+1.5V	J8.20	-12V

12.2.3 Unbuffered IO

The DN3000k10SD Daughter Card provides 66-unbuffered I/O signals, including 5 single ended clock signals available on headers J5, J6, and J7. The function of these signals is position dependent.

12.2.4 Buffered IO

The DN3000k10SD Daughter Card provides 48-buffered I/O signals available on headers J3, and J4. The function of these signals is position dependent. U1, U2, and U3 allow for different populating options, and devices can be active or passive:

Active - The LCV162245A is used for asynchronous communication between data buses. It allows data transmission from the A to the B or from the B to the A bus, depending on the logic level at the direction-control (DIR) input. The output-enable (OE#) input can be used to disable the device so that the busses are effectively isolated

Passive - The FST163245 bus switches are used to connect or isolate two ports without providing any current sink or source capabilities. Thus, they generate little or no noise of their own while providing a low resistance path for an external driver. The output-enable (OE#) input can be used to disable the device so that the busses are effectively isolated.

12.2.5 LVDS IO

Low-voltage differential signaling (LVDS) is a signaling method used for high-speed transmission of binary data over copper. It is well recognized that the benefits of balanced data transmission begin to outweigh the costs over single-ended techniques when the signal transmission times approach 10 ns. This represents signaling rates of about 30 Mbps or clock rates of 60 MHz (in single-edge clocking systems) and above. LVDS is defined in the TIA/EIA-644 standards.

Connector J1 is a Mini D Ribbon (MDR) connector (50-pin) manufactured by 3M, used specifically for high speed LVDS signaling. The connector mates with a standard off-the-shelf 3M-cable assembly:

P/N 14150-EZBB-XXX-0LC

where XXX is: 050 = 0.5 m

150 = 1.5 m

300 = 3.0 m

500 = 5.0 m

Please contact 3M for further details: <http://www.3m.com/>

12.2.6 Connection between FPGA and the Daughter Card Headers

Table 31 shows the IO connections between the DN3000K10SD headers and the FPGA IO pins. The VCCO of the IO banks are connected to +2.5V.

Table 31 - Connection between FPGA and the Daughter Card Headers

Daughter Card Connections			DN6000K10PCI IO Connections Test Header A		
Test Header	Signal Name	Connector	Test Header	Signal Name	FPGA Pin
J1.001	No Connect		P6.1	12V (+)	
J1.002	No Connect		P6.2	GND	
J1.003	ACLK1		P6.3	2.5V	
J1.004	No Connect	J5.1	P6.4	5V	
J1.005	BCLK1		P6.5	2.5V	
J1.006	No Connect	J5.3	P6.6	5V	
J1.007	CCLK1		P6.7	ACLK10	
J1.008	No Connect	J5.5	P6.8	GND	
J1.009	No Connect		P6.9	3.3V	
J1.010	BP2N3(P2N3)		P6.10	BCLK10	
J1.011	No Connect	J3.1	P6.11	GND	
J1.012	BP2N2(P2N2)		P6.12	TST_HDRA156	U16.AA34
J1.013	P2N1	J3.3	P6.13	TST_HDRA154	U16.AA37
J1.014	P2N0	J2.8	P6.14	TST_HDRA152	U16.AA31

Daughter Card Connections			DN6000K10PCI IO Connections Test Header A		
Test Header	Signal Name	Connector	Test Header	Signal Name	FPGA Pin
J1.015	BP2NX7(P2NX7)	J2.9	P6.15	TST_HDRA150	U16.AA40
J1.016	BP2NX6(P2NX6)	J3.5	P6.16	TST_HDRA148	U16.AB40
J1.017	BP2NX5(P2NX5)	J3.7	P6.17	TST_HDRA146	U16.AB37
J1.018	BP2NX4(P2NX4)	J3.9	P6.18	TST_HDRA144	U16.AB34
J1.019	P2NX1	J3.11	P6.19	TST_HDRA142	U16.AC40
J1.020	P2NX0	J2.10	P6.20	TST_HDRA140	U16.AC32
J1.021	P3NX9	J2.11	P6.21	TST_HDRA138	U16.AC37
J1.022	No Connect	J2.40	P6.22	GND	
J1.023	P3NX8		P6.23	TST_HDRA136	U16.AC34
J1.024	BP3NX5(P3NX5)	J2.41	P6.24	TST_HDRA134	U16.AD34
J1.025	BP3NX4(P3NX4)	J3.13	P6.25	TST_HDRA132	U16.AD42
J1.026	BP3N89(P3N89)	J3.15	P6.26	TST_HDRA130	U16.AD40
J1.027	BP3N88(P3N88)	J3.17	P6.27	TST_HDRA128	U16.AD32
J1.028	BP3N87(P3N87)	J3.19	P6.28	TST_HDRA126	U16.AD38
J1.029	BP3N86(P3N86)	J3.21	P6.29	TST_HDRA124	U16.AD36
J1.030	BP3N83(P3N83)	J3.23	P6.30	TST_HDRA122	U16.AE33
J1.031	BP3N82(P3N82)	J3.25	P6.31	TST_HDRA120	U16.AE42
J1.032	BP3N77(P3N77)	J3.27	P6.32	TST_HDRA118	U16.AE39

Daughter Card Connections			DN6000K10PCI IO Connections Test Header A		
Test Header	Signal Name	Connector	Test Header	Signal Name	FPGA Pin
J1.033	No Connect	J3.29	P6.33	GND	
J1.034	BP3N76(P3N76)		P6.34	TST_HDRA116	U16.AF32
J1.035	BP3N75(P3N75)	J3.31	P6.35	TST_HDRA114	U16.AE36
J1.036	BP3N74(P3N74)	J3.33	P6.36	TST_HDRA112	U16.AF42
J1.037	P3N69	J3.35	P6.37	TST_HDRA110	U16.AG31
J1.038	P3N68	J2.42	P6.38	TST_HDRA108	U16.AF40
J1.039	BP3N67(P3N67)	J2.43	P6.39	TST_HDRA106	U6.AF38
J1.040	BP3N66(P3N66)	J3.37	P6.40	TST_HDRA104	U6.AF36
J1.041	BP3N63(P3N63)	J3.39	P6.41	TST_HDRA102	U16.AF34
J1.042	BP3N62(P3N62)	J3.41	P6.42	TST_HDRA100	U15.AG41
J1.043	BP3N57(P3N57)	J3.43	P6.43	TST_HDRA98	U16.AG33
J1.044	No Connect	J3.45	P6.44	GND	
J1.045	BP3N56(P3N56)		P6.45	TST_HDRA96	U16.AG39
J1.046	No Connect	J3.47	P6.46	TST_HDRA94	U16.AG37
J1.047	No Connect		P6.47	TST_HDRA92	U16.AG35
J1.048	BP3N49(P3N49)	J4.1	P6.48	TST_HDRA90	U16.AH42
J1.049	BP3N48(P3N48)	J4.3	P6.49	TST_HDRA88	U16.AH40
J1.050	P3N47	J2.19	P6.50	TST_HDRA86	U16.AH32

Daughter Card Connections			DN6000K10PCI IO Connections Test Header A		
Test Header	Signal Name	Connector	Test Header	Signal Name	FPGA Pin
J1.051	P3N46	J2.20	P6.51	TST_HDRA84	U16.AH38
J1.052	BP3N43(P3N43)	J4.5	P6.52	TST_HDRA82	U16.AH34
J1.053	BP3N42(P3N42)	J4.7	P6.53	TST_HDRA80	U16.AJ32
J1.054	BP3N39(P3N39)	J4.9	P6.54	TST_HDRA78	U14.F36
J1.055	No Connect		P6.55	GND	
J1.056	BP3N38(P3N38)	J4.11	P6.56	TST_HDRA76	U14.E40
J1.057	BP3N35(P3N35)	J4.13	P6.57	TST_HDRA74	U14.D41
J1.058	BP3N34(P3N34)	J4.15	P6.58	TST_HDRA72	U14.E41
J1.059	BP3N29(P3N29)	J4.17	P6.59	TST_HDRA70	U14.F40
J1.060	BP3N28(P3N28)	J4.19	P6.60	TST_HDRA68	U14.F41
J1.061	BP3N27(P3N27)	J4.21	P6.61	TST_HDRA66	14.G39
J1.062	BP3N26(P3N26)	J4.23	P6.62	TST_HDRA64	U14.G42
J1.063	P3N23	J2.21	P6.63	TST_HDRA62	U14.H37
J1.064	P3N22	J2.22	P6.64	TST_HDRA60	U14.H38
J1.065	BP3N19(P3N19)	J4.25	P6.65	TST_HDRA58	U14.H41
J1.066	No Connect		P6.66	GND	
J1.067	BP3N18(P3N18)	J4.27	P6.67	TST_HDRA56	U14.J35
J1.068	BP3N15(P3N15)	J4.29	P6.68	TST_HDRA54	U14.J36

Daughter Card Connections			DN6000K10PCI IO Connections Test Header A		
Test Header	Signal Name	Connector	Test Header	Signal Name	FPGA Pin
J1.069	BP3N14(P3N14)	J4.31	P6.69	TST_HDRA52	U14.J39
J1.070	P3N9	J2.23	P6.70	TST_HDRA50	U14.J41
J1.071	P3N8	J2.24	P6.71	TST_HDRA48	U14.L33
J1.072	BP3N7(P3N7)	J4.33	P6.72	TST_HDRA46	U14.K38
J1.073	BP3N6(P3N6)	J4.35	P6.73	TST_HDRA44	U14.K36
J1.074	BP3N3(P3N3)	J4.37	P6.74	TST_HDRA42	U14.K42
J1.075	BP3N2(P3N2)	J4.39	P6.75	TST_HDRA40	U14.L34
J1.076	BP4N27(P4N27)	J4.41	P6.76	TST_HDRA38	U14.L36
J1.077	No Connect		P6.77	GND	
J1.078	BP4N26(P4N26)	J4.43	P6.78	TST_HDRA36	U14.L40
J1.079	BP4N21(P4N21)	J4.45	P6.79	TST_HDRA34	U14.L39
J1.080	BP4N20(P4N20)	J4.47	P6.80	TST_HDRA32	U14.L41
J1.081	No Connect		P6.81	TST_HDRA30	U14.M35
J1.082	No Connect		P6.82	TST_HDRA28	U14.M31
J1.083	No Connect		P6.83	TST_HDRA26	U14.M33
J1.084	No Connect		P6.84	TST_HDRA24	U14.M40
J1.085	No Connect		P6.85	TST_HDRA22	U14.M38
J1.086	No Connect		P6.86	TST_HDRA20	U14.N35

Daughter Card Connections			DN6000K10PCI IO Connections Test Header A		
Test Header	Signal Name	Connector	Test Header	Signal Name	FPGA Pin
J1.087	No Connect		P6.87	TST_HDRA18	U14.N37
J1.088	No Connect		P6.88	GND	
J1.089	No Connect		P6.89	TST_HDRA16	U14.N33
J1.090	No Connect		P6.90	TST_HDRA14	U14.N39
J1.091	No Connect		P6.91	TST_HDRA12	U14.N41
J1.092	No Connect		P6.92	TST_HDRA10	U14.N31
J1.093	No Connect		P6.93	1.5V	
J1.094	No Connect		P6.94	TST_HDRA8	U14.P33
J1.095	P4NX7	J7.45	P6.95	TST_HDRA6	U14.P35
J1.096	P4NX6	J7.47	P6.96	TST_HDRA4	U14.P31
J1.097	No Connect		P6.97	TST_HDRA2	U14.P37
J1.098	No Connect		P6.98	TST_HDRA0	U14.P41
J1.099	No Connect		P6.99	GND	
J1.100	No Connect		P6.100	12V (-)	
J1.101	No Connect		P6.101	GND	
J1.102	MBCK1	J2.27	P6.102	TST_HDRA_CL KIN	U14.AT22
J1.103	No Connect		P6.103	1.5V	

Daughter Card Connections			DN6000K10PCI IO Connections Test Header A		
Test Header	Signal Name	Connector	Test Header	Signal Name	FPGA Pin
J1.104	MBCK0	J2.28	P6.104	GND	
J1.105	No Connect		P6.105	3.3V	
J1.107	No Connect		P6.107	GND	
J1.108	ECLK1	J5.7	P6.108	GND	
J1.109	No Connect		P6.109	GND	
J1.110	No Connect		P6.110	GND	
J1.111	P2N5	J5.15	P6.111	TST_HDRA160	U16.Y40
J1.112	P2N4	J5.17	P6.112	TST_HDRA159	U16.Y39
J1.113	P2NX11	J2.2	P6.113	TST_HDRA158	U16.Y32
J1.114	P2NX10	J2.1	P6.114	TST_HDRA157	U16.Y31
J1.115	P2NX9	J5.19	P6.115	TST_HDRA155	U16.AA33
J1.116	P2NX8	J5.21	P6.116	TST_HDRA153	U16.AA36
J1.117	P2NX3	J5.23	P6.117	TST_HDRA151	U16.AB31
J1.118	No Connect		P6.118	GND	
J1.119	P2NX2	J5.25	P6.119	TST_HDRA149	U16.AA39
J1.120	P3NX11	J2.29	P6.120	TST_HDRA147	U16.AB39
J1.121	P3NX10	J2.30	P6.121	TST_HDRA145	U16.AB36
J1.122	P3NX7	J2.31	P6.122	TST_HDRA143	U16.AB33

Daughter Card Connections			DN6000K10PCI IO Connections Test Header A		
Test Header	Signal Name	Connector	Test Header	Signal Name	FPGA Pin
J1.123	P3NX6	J2.32	P6.123	TST_HDRA141	U16.AC39
J1.124	P3NX3	J5.27	P6.124	TST_HDRA139	U16.AC31
J1.125	P3NX2	J5.29	P6.125	TST_HDRA137	U16.AC36
J1.126	P3NX1	J5.31	P6.126	TST_HDRA135	U16.AC33
J1.127	P3NX0	J5.33	P6.127	TST_HDRA133	U16.AD33
J1.128	P3N85	J5.35	P6.128	TST_HDRA131	U16.AD41
J1.129	No Connect		P6.129	GND	
J1.130	P3N84	J5.37	P6.130	TST_HDRA129	U16.AD39
J1.131	P3N81	J5.39	P6.131	TST_HDRA127	U16.AD31
J1.132	P3N80	J5.41	P6.132	TST_HDRA125	U16.AD37
J1.133	P3N79	J2.3	P6.133	TST_HDRA123	U16.AD35
J1.134	P3N78	J2.4	P6.134	TST_HDRA121	U16.AE32
J1.135	P3N73	J2.6	P6.135	TST_HDRA119	U16.AE41
J1.136	P3N72	J2.7	P6.136	TST_HDRA117	U16.AE38
J1.137	P3N71	J2.33	P6.137	TST_HDRA115	U16.AE31
J1.138	P3N70	J2.34	P6.138	TST_HDRA113	U16.AE35
J1.139	P3N65	J5.43	P6.139	TST_HDRA111	U16.AF41
J1.140	No Connect		P6.140	GND	

Daughter Card Connections			DN6000K10PCI IO Connections Test Header A		
Test Header	Signal Name	Connector	Test Header	Signal Name	FPGA Pin
J1.141	P3N64	J5.45	P6.141	TST_HDRA109	U16.AF31
J1.142	P3N61	J5.47	P6.142	TST_HDRA107	U16.AF39
J1.143	P3N60	J5.49	P6.143	TST_HDRA105	U6.AF37
J1.144	P3N59	J6.1	P6.144	TST_HDRA103	U6.AF35
J1.145	P3N58	J6.3	P6.145	TST_HDRA101	U16.AF33
J1.146	P3N53	J6.5	P6.146	TST_HDRA99	U16.AG40
J1.147	P3N52	J6.7	P6.147	TST_HDRA97	U16.AD32
J1.148	P3N51	J2.17	P6.148	TST_HDRA95	U16.AG38
J1.149	P3N50	J2.18	P6.149	TST_HDRA93	U16.AG36
J1.150	P3N45	J6.9	P6.150	TST_HDRA91	U16.AH35
J1.151	No Connect		P6.151	GND	
J1.152	P3N44	J6.11	P6.152	TST_HDRA89	U16.AH41
J1.153	P3N41	J6.13	P6.153	TST_HDRA87	U16.AJ40
J1.154	P3N40	J6.15	P6.154	TST_HDRA85	U16.AH31
J1.155	P3N37	J6.17	P6.155	TST_HDRA83	U16.AH37
J1.156	P3N36	J6.19	P6.156	TST_HDRA81	U16.AH33
J1.157	P3N33	J6.21	P6.157	TST_HDRA79	U16.AJ31
J1.158	P3N32	J6.23	P6.158	TST_HDRA77	U14.D40

Daughter Card Connections			DN6000K10PCI IO Connections Test Header A		
Test Header	Signal Name	Connector	Test Header	Signal Name	FPGA Pin
J1.159	P3N31	J2.44	P6.159	TST_HDRA75	U14.D42
J1.160	P3N30	J2.45	P6.160	TST_HDRA73	U14.E42
J1.161	P3N25	J6.25	P6.161	TST_HDRA71	U14.F39
J1.162	No Connect		P6.162	GND	
J1.163	P3N24	J6.27	P6.163	TST_HDRA69	U14.F42
J1.164	P3N21	J6.29	P6.164	TST_HDRA67	U14.G40
J1.165	P3N20	J6.31	P6.165	TST_HDRA65	U14.T41
J1.166	P3N17	J6.33	P6.166	TST_HDRA63	U14.G38
J1.167	P3N16	J6.35	P6.167	TST_HDRA61	U14.H39
J1.168	P3N13	J6.37	P6.168	TST_HDRA59	U14.H40
J1.169	P3N12	J6.39	P6.169	TST_HDRA57	U14.H36
J1.170	P3N11	J2.47	P6.170	TST_HDRA55	U14.J37
J1.171	P3N10	J2.48	P6.171	TST_HDRA53	U14.J38
J1.172	P3N5	J6.41	P6.172	TST_HDRA51	U14.J42
J1.173	No Connect		P6.173	GND	
J1.174	P3N4	J6.43	P6.174	TST_HDRA49	U14.K34
J1.175	P3N1	J6.45	P6.175	TST_HDRA47	U14.K37
J1.176	P3N0	J6.47	P6.176	TST_HDRA45	U14.K35

Daughter Card Connections			DN6000K10PCI IO Connections Test Header A		
Test Header	Signal Name	Connector	Test Header	Signal Name	FPGA Pin
J1.177	P4N25	J7.1	P6.177	TST_HDRA43	U14.K41
J1.178	P4N24	J7.3	P6.178	TST_HDRA41	U14.L35
J1.179	P4N23	J7.5	P6.179	TST_HDRA39	U14.L37
J1.180	P4N22	J7.7	P6.180	TST_HDRA37	U14.K40
J1.181	P4N17	J7.9	P6.181	TST_HDRA35	U14.L38
J1.182	P4N16	J7.11	P6.182	TST_HDRA33	U14.L42
J1.183	P4N15	J7.13	P6.183	TST_HDRA31	U14.M36
J1.184	GND	J2.36	P6.184	GND	
J1.185	P4N14	J7.15	P6.185	TST_HDRA29	U14.M32
J1.186	P4N9	J7.17	P6.186	TST_HDRA27	U14.M34
J1.187	P4N8	J7.19	P6.187	TST_HDRA25	U14.M41
J1.188	P4N5	J7.21	P6.188	TST_HDRA23	U14.M39
J1.189	P4N4	J7.23	P6.189	TST_HDRA21	U14.N36
J1.190	P4N1	J7.25	P6.190	TST_HDRA19	U14.N38
J1.191	P4N0	J7.27	P6.191	TST_HDRA17	U14.N34
J1.192	P4NX13	J7.29	P6.192	TST_HDRA15	U4.N40
J1.193	P4NX12	J7.31	P6.193	TST_HDRA13	U16.N42
J1.194	P4NX9	J7.33	P6.194	TST_HDRA11	U14.N32

Daughter Card Connections			DN6000K10PCI IO Connections Test Header A		
Test Header	Signal Name	Connector	Test Header	Signal Name	FPGA Pin
J1.195	No Connect		P6.195	GND	
J1.196	P4NX8	J7.35	P6.196	TST_HDRA9	U14.P34
J1.197	P4NX3	J7.37	P6.197	TST_HDRA7	U14.P36
J1.198	P4NX2	J7.39	P6.198	TST_HDRA5	U14.P32
J1.199	P4NX1	J7.41	P6.199	TST_HDRA3	U14.P38
J1.200	P4NX0	J7.43	P6.200	TST_HDRA1	U14.P42

Daughter Card Connections			DN6000K10PCI IO Connections Test Header B		
Test Header	Signal Name	Connector	Test Header	Signal Name	FPGA Pin
J1.001	No Connect		P8.1	12V (+)	
J1.002	No Connect		P8.2	GND	
J1.003	ACLK1	J5.1	P8.3	2.5V	
J1.004	No Connect		P8.4	5V	
J1.005	BCLK1	J5.3	P8.5	2.5V	
J1.006	No Connect		P8.6	5V	
J1.007	CCLK1	J5.5	P8.7	ACLK11	
J1.008	No Connect		P8.8	GND	
J1.009	No Connect		P8.9	3.3V	
J1.010	BP2N3(P2N3)	J3.1	P8.10	BCLK11	
J1.011	No Connect		P8.11	GND	
J1.012	BP2N2(P2N2)	J3.3	P8.12	TST_HDRB0	U55.N1
J1.013	P2N1	J2.8	P8.13	TST_HDRB2	U55.E2
J1.014	P2N0	J2.9	P8.14	TST_HDRB4	U55.P9
J1.015	BP2NX7(P2NX7)	J3.5	P8.15	TST_HDRB6	U55.P7
J1.016	BP2NX6(P2NX6)	J3.7	P8.16	TST_HDRB8	U55.P11
J1.017	BP2NX5(P2NX5)	J3.9	P8.17	TST_HDRB10	U55.P5

Daughter Card Connections			DN6000K10PCI IO Connections Test Header B		
Test Header	Signal Name	Connector	Test Header	Signal Name	FPGA Pin
J1.018	BP2NX4(P2NX4)	J3.11	P8.18	TST_HDRB12	U55.P1
J1.019	P2NX1	J2.10	P8.19	TST_HDRB14	U55.R9
J1.020	P2NX0	J2.11	P8.20	TST_HDRB16	U55.R6
J1.021	P3NX9	J2.40	P8.21	TST_HDRB18	U55.R3
J1.022	No Connect		P8.22	GND	
J1.023	P3NX8	J2.41	P8.23	TST_HDRB20	U55.R2
J1.024	BP3NX5(P3NX5)	J3.13	P8.24	TST_HDRB22	U55.R12
J1.025	BP3NX4(P3NX4)	J3.15	P8.25	TST_HDRB24	U55.T7
J1.026	BP3N89(P3N89)	J3.17	P8.26	TST_HDRB26	U55.R8
J1.027	BP3N88(P3N88)	J3.19	P8.27	TST_HDRB28	U55.T5
J1.028	BP3N87(P3N87)	J3.21	P8.28	TST_HDRB30	U55.T3
J1.029	BP3N86(P3N86)	J3.23	P8.29	TST_HDRB32	55.T11
J1.030	BP3N83(P3N83)	J3.25	P8.30	TST_HDRB34	U55.U8
J1.031	BP3N82(P3N82)	J3.27	P8.31	TST_HDRB36	U55.U6
J1.032	BP3N77(P3N77)	J3.29	P8.32	TST_HDRB38	U55.U10
J1.033	No Connect		P8.33	GND	
J1.034	BP3N76(P3N76)	J3.31	P8.34	TST_HDRB40	U55.U4
J1.035	BP3N75(P3N75)	J3.33	P8.35	TST_HDRB42	U55.U2

Daughter Card Connections			DN6000K10PCI IO Connections Test Header B		
Test Header	Signal Name	Connector	Test Header	Signal Name	FPGA Pin
J1.036	BP3N74(P3N74)	J3.35	P8.36	TST_HDRB44	U55.U12
J1.037	P3N69	J2.42	P8.37	TST_HDRB46	U55.V11
J1.038	P3N68	J2.43	P8.38	TST_HDRB48	U55.V8
J1.039	BP3N67(P3N67)	J3.37	P8.39	TST_HDRB50	U55.V12
J1.040	BP3N66(P3N66)	J3.39	P8.40	TST_HDRB52	U55.V5
J1.041	BP3N63(P3N63)	J3.41	P8.41	TST_HDRB54	U55.V2
J1.042	BP3N62(P3N62)	J3.43	P8.42	TST_HDRB56	U55.W10
J1.043	BP3N57(P3N57)	J3.45	P8.43	TST_HDRB58	U55.W8
J1.044	No Connect		P8.44	GND	
J1.045	BP3N56(P3N56)	J3.47	P8.45	TST_HDRB60	U55.W6
J1.046	No Connect		P8.46	TST_HDRB62	U55.W12
J1.047	No Connect		P8.47	TST_HDRB64	U55.W4
J1.048	BP3N49(P3N49)	J4.1	P8.48	TST_HDRB66	U55.W2
J1.049	BP3N48(P3N48)	J4.3	P8.49	TST_HDRB68	U55.Y10
J1.050	P3N47	J2.19	P8.50	TST_HDRB70	U55.Y7
J1.051	P3N46	J2.20	P8.51	TST_HDRB72	U55.Y4
J1.052	BP3N43(P3N43)	J4.5	P8.52	TST_HDRB74	U55.Y12
J1.053	BP3N42(P3N42)	J4.7	P8.53	TST_HDRB76	U55.AA10

Daughter Card Connections			DN6000K10PCI IO Connections Test Header B		
Test Header	Signal Name	Connector	Test Header	Signal Name	FPGA Pin
J1.054	BP3N39(P3N39)	J4.9	P8.54	TST_HDRB78	U55.AA7
J1.055	No Connect		P8.55	GND	
J1.056	BP3N38(P3N38)	J4.11	P8.56	TST_HDRB80	U54.AJ2
J1.057	BP3N35(P3N35)	J4.13	P8.57	TST_HDRB82	U54.AJ6
J1.058	BP3N34(P3N34)	J4.15	P8.58	TST_HDRB84	U54.AJ10
J1.059	BP3N29(P3N29)	J4.17	P8.59	TST_HDRB86	U43.AJ8
J1.060	BP3N28(P3N28)	J4.19	P8.60	TST_HDRB88	U54.AK2
J1.061	BP3N27(P3N27)	J4.21	P8.61	TST_HDRB90	U54.AK12
J1.062	BP3N26(P3N26)	J4.23	P8.62	TST_HDRB92	U54.AJ4
J1.063	P3N23	J2.21	P8.63	TST_HDRB94	U54.AK6
J1.064	P3N22	J2.22	P8.64	TST_HDRB96	U54.AK10
J1.065	BP3N19(P3N19)	J4.25	P8.65	TST_HDRB98	U54.AK8
J1.066	No Connect		P8.66	GND	
J1.067	BP3N18(P3N18)	J4.27	P8.67	TST_HDRB100	U54.AL3
J1.068	BP3N15(P3N15)	J4.29	P8.68	TST_HDRB102	U54.AL12
J1.069	BP3N14(P3N14)	J4.31	P8.69	TST_HDRB104	U54.AL5
J1.070	P3N9	J2.23	P8.70	TST_HDRB106	U54.AL8
J1.071	P3N8	J2.24	P8.71	TST_HDRB108	U54.AL10

Daughter Card Connections			DN6000K10PCI IO Connections Test Header B		
Test Header	Signal Name	Connector	Test Header	Signal Name	FPGA Pin
J1.072	BP3N7(P3N7)	J4.33	P8.72	TST_HDRB110	U54.AM2
J1.073	BP3N6(P3N6)	J4.35	P8.73	TST_HDRB112	U54.AN3
J1.074	BP3N3(P3N3)	J4.37	P8.74	TST_HDRB114	U54.AM9
J1.075	BP3N2(P3N2)	J4.39	P8.75	TST_HDRB116	U54.AM5
J1.076	BP4N27(P4N27)	J4.41	P8.76	TST_HDRB118	U54.AM7
J1.077	No Connect		P8.77	GND	
J1.078	BP4N26(P4N26)	J4.43	P8.78	TST_HDRB120	U54.AM10
J1.079	BP4N21(P4N21)	J4.45	P8.79	TST_HDRB122	U54.AN2
J1.080	BP4N20(P4N20)	J4.47	P8.80	TST_HDRB124	U54.AN6
J1.081	No Connect		P8.81	TST_HDRB126	U54.AN8
J1.082	No Connect		P8.82	TST_HDRB128	U54.AP2
J1.083	No Connect		P8.83	TST_HDRB130	U54.AP5
J1.084	No Connect		P8.84	TST_HDRB132	U54.AP8
J1.085	No Connect		P8.85	TST_HDRB134	U54.AP7
J1.086	No Connect		P8.86	TST_HDRB136	U54.AR3
J1.087	No Connect		P8.87	TST_HDRB138	U54.AR6
J1.088	No Connect		P8.88	GND	
J1.089	No Connect		P8.89	TST_HDRB140	U54.AR5

Daughter Card Connections			DN6000K10PCI IO Connections Test Header B		
Test Header	Signal Name	Connector	Test Header	Signal Name	FPGA Pin
J1.090	No Connect		P8.90	TST_HDRB142	U54.AT2
J1.091	No Connect		P8.91	TST_HDRB144	U54.AT4
J1.092	No Connect		P8.92	TST_HDRB146	U54.AU2
J1.093	No Connect		P8.93	1.5V	
J1.094	No Connect		P8.94	TST_HDRB148	U54.AU4
J1.095	P4NX7	J7.45	P8.95	TST_HDRB150	U54.AW3
J1.096	P4NX6	J7.47	P8.96	TST_HDRB152	U54.AV2
J1.097	No Connect		P8.97	TST_HDRB154	U54.AW2
J1.098	No Connect		P8.98	TST_HDRB156	U54.AU8
J1.099	No Connect		P8.99	GND	
J1.100	No Connect		P8.100	12V (-)	
J1.101	No Connect		P8.101	GND	
J1.102	MBCK1	J2.27	P8.102	TST_HDRB_CL KIN	U55.AN22
J1.103	No Connect		P8.103	1.5V	
J1.104	MBCK0	J2.28	P8.104	GND	
J1.105	No Connect		P8.105	3.3V	
J1.107	No Connect		P8.107	GND	

Daughter Card Connections			DN6000K10PCI IO Connections Test Header B		
Test Header	Signal Name	Connector	Test Header	Signal Name	FPGA Pin
J1.108	ECLK1	J5.7	P8.108	GND	
J1.109	No Connect		P8.109	GND	
J1.110	No Connect		P8.110	GND	
J1.111	P2N5	J5.15	P8.111	TST_HDRB1	U55.N2
J1.112	P2N4	J5.17	P8.112	TST_HDRB3	U55.N12
J1.113	P2NX11	J2.2	P8.113	TST_HDRB5	U55.P10
J1.114	P2NX10	J2.1	P8.114	TST_HDRB7	U55.P8
J1.115	P2NX9	J5.19	P8.115	TST_HDRB9	U54.P12
J1.116	P2NX8	J5.21	P8.116	TST_HDRB11	U55.P6
J1.117	P2NX3	J5.23	P8.117	TST_HDRB13	U55.P2
J1.118	No Connect		P8.118	GND	
J1.119	P2NX2	J5.25	P8.119	TST_HDRB15	U55.R10
J1.120	P3NX11	J2.29	P8.120	TST_HDRB17	U55.P3
J1.121	P3NX10	J2.30	P8.121	TST_HDRB19	U55.N11
J1.122	P3NX7	J2.31	P8.122	TST_HDRB21	U55.R11
J1.123	P3NX6	J2.32	P8.123	TST_HDRB23	U55.T6
J1.124	P3NX3	J5.27	P8.124	TST_HDRB25	U55.T8
J1.125	P3NX2	J5.29	P8.125	TST_HDRB27	U55.T4

Daughter Card Connections			DN6000K10PCI IO Connections Test Header B		
Test Header	Signal Name	Connector	Test Header	Signal Name	FPGA Pin
J1.126	P3NX1	J5.31	P8.126	TST_HDRB29	U55.T2
J1.127	P3NX0	J5.33	P8.127	TST_HDRB31	U55.T10
J1.128	P3N85	J5.35	P8.128	TST_HDRB33	55.U7
J1.129	No Connect		P8.129	GND	
J1.130	P3N84	J5.37	P8.130	TST_HDRB35	U55.U5
J1.131	P3N81	J5.39	P8.131	TST_HDRB37	U55.U9
J1.132	P3N80	J5.41	P8.132	TST_HDRB39	U55.U3
J1.133	P3N79	J2.3	P8.133	TST_HDRB41	U55.U1
J1.134	P3N78	J2.4	P8.134	TST_HDRB43	U55.T12
J1.135	P3N73	J2.6	P8.135	TST_HDRB45	U55.V10
J1.136	P3N72	J2.7	P8.136	TST_HDRB47	U55.V7
J1.137	P3N71	J2.33	P8.137	TST_HDRB49	U55.U11
J1.138	P3N70	J2.34	P8.138	TST_HDRB51	U55.V4
J1.139	P3N65	J5.43	P8.139	TST_HDRB53	U55.V1
J1.140	No Connect		P8.140	GND	
J1.141	P3N64	J5.45	P8.141	TST_HDRB55	U55.W9
J1.142	P3N61	J5.47	P8.142	TST_HDRB57	U55.W7
J1.143	P3N60	J5.49	P8.143	TST_HDRB59	U55.W5

Daughter Card Connections			DN6000K10PCI IO Connections Test Header B		
Test Header	Signal Name	Connector	Test Header	Signal Name	FPGA Pin
J1.144	P3N59	J6.1	P8.144	TST_HDRB61	U55.W11
J1.145	P3N58	J6.3	P8.145	TST_HDRB63	U55.W3
J1.146	P3N53	J6.5	P8.146	TST_HDRB65	U55.W1
J1.147	P3N52	J6.7	P8.147	TST_HDRB67	U55.Y9
J1.148	P3N51	J2.17	P8.148	TST_HDRB69	U55.Y6
J1.149	P3N50	J2.18	P8.149	TST_HDRB71	U55.Y3
J1.150	P3N45	J6.9	P8.150	TST_HDRB73	U55.Y11
J1.151	No Connect		P8.151	GND	
J1.152	P3N44	J6.11	P8.152	TST_HDRB75	U55.AA9
J1.153	P3N41	J6.13	P8.153	TST_HDRB77	U55.AA6
J1.154	P3N40	J6.15	P8.154	TST_HDRB79	U54.AJ1
J1.155	P3N37	J6.17	P8.155	TST_HDRB81	U54.AJ5
J1.156	P3N36	J6.19	P8.156	TST_HDRB83	U54.AJ9
J1.157	P3N33	J6.21	P8.157	TST_HDRB85	U54.AJ7
J1.158	P3N32	J6.23	P8.158	TST_HDRB87	U54.AK1
J1.159	P3N31	J2.44	P8.159	TST_HDRB89	U54.AK11
J1.160	P3N30	J2.45	P8.160	TST_HDRB91	U54.AK3
J1.161	P3N25	J6.25	P8.161	TST_HDRB93	U54.AK5

Daughter Card Connections			DN6000K10PCI IO Connections Test Header B		
Test Header	Signal Name	Connector	Test Header	Signal Name	FPGA Pin
J1.162	No Connect		P8.162	GND	
J1.163	P3N24	J6.27	P8.163	TST_HDRB95	U54.AK9
J1.164	P3N21	J6.29	P8.164	TST_HDRB97	U54.AK7
J1.165	P3N20	J6.31	P8.165	TST_HDRB99	U54.AL2
J1.166	P3N17	J6.33	P8.166	TST_HDRB101	U54.AL11
J1.167	P3N16	J6.35	P8.167	TST_HDRB103	U54.AL4
J1.168	P3N13	J6.37	P8.168	TST_HDRB105	U54.AL7
J1.169	P3N12	J6.39	P8.169	TST_HDRB107	U54.AL9
J1.170	P3N11	J2.47	P8.170	TST_HDRB109	U54.AM1
J1.171	P3N10	J2.48	P8.171	TST_HDRB111	U54.AM3
J1.172	P3N5	J6.41	P8.172	TST_HDRB113	U54.AM8
J1.173	No Connect		P8.173	GND	
J1.174	P3N4	J6.43	P8.174	TST_HDRB115	U54.AM4
J1.175	P3N1	J6.45	P8.175	TST_HDRB117	U54.AM6
J1.176	P3N0	J6.47	P8.176	TST_HDRB119	U54.AN9
J1.177	P4N25	J7.1	P8.177	TST_HDRB121	U54.AN1
J1.178	P4N24	J7.3	P8.178	TST_HDRB123	U54.AN5
J1.179	P4N23	J7.5	P8.179	TST_HDRB125	U54.AN7

Daughter Card Connections			DN6000K10PCI IO Connections Test Header B		
Test Header	Signal Name	Connector	Test Header	Signal Name	FPGA Pin
J1.180	P4N22	J7.7	P8.180	TST_HDRB127	U54.AP1
J1.181	P4N17	J7.9	P8.181	TST_HDRB129	U54.AP4
J1.182	P4N16	J7.11	P8.182	TST_HDRB131	U54.AR7
J1.183	P4N15	J7.13	P8.183	TST_HDRB133	U54.AP6
J1.184	GND	J2.36	P8.184	GND	
J1.185	P4N14	J7.15	P8.185	TST_HDRB135	U54.AR2
J1.186	P4N9	J7.17	P8.186	TST_HDRB137	U54.AT5
J1.187	P4N8	J7.19	P8.187	TST_HDRB139	U54.AR4
J1.188	P4N5	J7.21	P8.188	TST_HDRB141	U54.AT1
J1.189	P4N4	J7.23	P8.189	TST_HDRB143	U54.AT3
J1.190	P4N1	J7.25	P8.190	TST_HDRB145	U54.AU1
J1.191	P4N0	J7.27	P8.191	TST_HDRB147	U54.AU3
J1.192	P4NX13	J7.29	P8.192	TST_HDRB149	U54.AV3
J1.193	P4NX12	J7.31	P8.193	TST_HDRB151	U54.AV1
J1.194	P4NX9	J7.33	P8.194	TST_HDRB153	U54.AW1
J1.195	No Connect		P8.195	GND	
J1.196	P4NX8	J7.35	P8.196	TST_HDRB155	U54.AT8
J1.197	P4NX3	J7.37	P8.197	TST_HDRB157	U54.AT6

Daughter Card Connections			DN6000K10PCI IO Connections Test Header B		
Test Header	Signal Name	Connector	Test Header	Signal Name	FPGA Pin
J1.198	P4NX2	J7.39	P8.198	TST_HDRB158	U54.AU7
J1.199	P4NX1	J7.41	P8.199	TST_HDRB159	U54.AY5

13 Mechanical

Two bus bars, MP1 and MP3 are installed to prevent flexing of the PWB. They are connected to the ground plane and can be used to ground test equipment. The user must not short any power rails or signals to these metal bars - they can conduct a lot of current. Mounting holes are provided to allow the PCB to be mounted in a case.

13.1.1 PWB Dimension

The DN6000K10PCI PWB conforms to the following dimensions:



Appendix A – Address Maps

The DN6000k10PCI reference design can be used to verify the functionality of the board. There are several ways to exercise the reference design features. In each FPGA there is PowerPC code that allows the user to communicate directly with the FPGA through the RS232 port that is discussed in [Using the Reference Design](#). Another method of communication is through USB (J4) or the Cypress MCU RS232 port P2. The USB PC application can be found on the product CD in “Source Code\USBController\USBController.exe”. This application allows the user to read/write to different FPGA addresses and also perform tests on the DDR, SRAM, internal registers, and interconnect between the FPGA’s ([Description of Main Menu Options](#)). The following 6 tables are the address maps for each FPGA when communicating through USB or via the RS232 port on the MCU (P7). Please note these address maps are not the same for communication through the PPC RS232 port menus (please see [Using the Reference Design](#) for a description). Also note that The Dini Group reference design provided with the DN6000k10PCI must be loaded in each of the existing FPGA’s for the following address maps to be valid.

The 32-bit address space is decoded as follows:

- Bits[31:28] – select an FPGA (A = 0, B = 1, C = 2, ..., F = 5)
- Bit 27 - selects between DDR (0) and SRAM/REGISTERS (1)
- Bit 26 – selects between SRAM(0) and REGISTERS (1) if bit 27 is 1
- Bit 25 – Select ddr1(0) or ddr2(1) if the 128MB ddr parts are stuffed
- Bit 24 – selects ddr1(0) or ddr2(1) if the 64MB ddr parts are stuffed

FPGA A

	Start Address	End Address	Read / Write	Description
SRAM (U168)	0x0800_0000	0x0807_FFFF	R/W	Address maps directly to SRAM (512k x 36). If larger SRAMs are installed then the address space just extends upward.
External Host Commands Register	0x0C00_0004	0x0C00_0004	R/W	Write/Read register for MCU to issue the following commands: 0x1 – test all functionality 0x2 – test registers 0x3 – test SRAM 0x5 – test FPGA interconnect To issue a command the MCU must write one of the above values to this register. The MCU can then poll this register to check if the test is done (register will return all zeros when finished)
Status Register	0x0C00_0008	0x0C00_0008	R	Read-only register - Status of commands and bus control: Bits 16-18 must all be zero for MCU to issue any commands to External Host Command Register. Status results can be read after command register is cleared. The decode for the test results is as follows: Bit 0 – overall pass/fail (pass = 1, fail = 0) Bit 1 – register test pass/fail Bit 2 – flash test pass/fail Bit 3 – ddr test pass/fail Bit 4 – interconnect test pass/fail
Existing FPGA Register	0x0C00_0018	0x0C00_0018	R/W	Contains information on what FPGAs are stuffed on the DN6000K10PCI as well as what type of FPGAs they are. The register has the following format: Bit 0 – 1 if FPGA A is stuffed, 0 otherwise Bit 1 – 1 if FPGA B is stuffed, 0 otherwise Bit 2 – 1 if FPGA C is stuffed, 0 otherwise Bit 2 – 1 if FPGA D is stuffed, 0 otherwise Bit 2 – 1 if FPGA E is stuffed, 0 otherwise Bit 2 – 1 if FPGA F is stuffed, 0 otherwise Bit 2 – 1 if FPGA G is stuffed, 0 otherwise Bit 7 – 1 if FPGA H is stuffed, 0 otherwise Bit 8 – 1 if FPGA I is stuffed, 0 otherwise Bit 9 – 1 if FPGA A is 2vp100, 0 if 2vp70 Bit 10 – 1 if FPGA B is 2vp100, 0 if 2vp70 Bit 11 – 1 if FPGA C is 2vp100, 0 if 2vp70 Bit 12 – 1 if FPGA D is 2vp100, 0 if 2vp70 Bit 13 – 1 if FPGA E is 2vp100, 0 if 2vp70 Bit 14 – 1 if FPGA F is 2vp100, 0 if 2vp70 Bit 15 – 1 if FPGA G is 2vp100, 0 if 2vp70 Bit 16 – 1 if FPGA H is 2vp100, 0 if 2vp70 Bit 17 – 1 if FPGA I is 2vp100, 0 if 2vp70

FPGA B

	Start Address	End Address	Read / Write	Description
DDR 1 (U11)	0x1000_0000	0x10FF_FFFF	R/W	Address maps directly to DDR 1 (32Mx16). If larger DDRs are installed then the address space just extends upward and the start address for DDR2 is moved upward by the same amount.
DDR 2 (U19)	0x1100_0000	0x11FF_FFFF	R/W	Address maps directly to DDR 2 (U34 only avail if 2vp100). If larger DDRs are installed then the address space just extends upward.
SRAM (U69)	0x1800_0000	0x1807_FFFF	R/W	Address maps directly to SRAM. If larger SRAMs are installed then the address space just extends upward.
DDR Phase Shift Register	0x1C00_0000	0x1C00_0000	R/W	DDR phase shift value (upper WORD is read only and contains the current phase shift value, lower WORD is write only)
External Host Commands Register	0x1C00_0004	0x1C00_0004	R/W	Write/Read register for MCU to issue the following commands: 0x1 – test all functionality 0x2 – test registers 0x3 – test SRAM 0x4 – test DDR(s) 0x5 – test FPGA interconnect To issue a command the MCU must write one of the above values to this register. The MCU can then poll this register to check if the test is done (register will return all zeros when finished)
Status Register	0x1C00_0008	0x1C00_0008	R	Read-only register - Status of commands and bus control: Bits 16-18 must all be zero for MCU to issue any commands to External Host Command Register. Status results can be read after command register is cleared. The decode for the test results is as follows: Bit 0 – overall pass/fail (pass = 1, fail = 0) Bit 1 – register test pass/fail Bit 2 – flash test pass/fail Bit 3 – ddr test pass/fail Bit 4 – interconnect test pass/fail
Existing FPGA Register	0x1C00_0018	0x1C00_0018	R/W	Contains information on what FPGAs are stuffed on the DN6000k10PCI as well as what type of FPGAs they are. The register has the following format: Bit 0 – 1 if FPGA A is stuffed, 0 otherwise Bit 1 – 1 if FPGA B is stuffed, 0 otherwise Bit 2 – 1 if FPGA C is stuffed, 0 otherwise Bit 2 – 1 if FPGA D is stuffed, 0 otherwise Bit 2 – 1 if FPGA E is stuffed, 0 otherwise Bit 2 – 1 if FPGA F is stuffed, 0 otherwise Bit 2 – 1 if FPGA G is stuffed, 0 otherwise Bit 7 – 1 if FPGA H is stuffed, 0 otherwise Bit 8 – 1 if FPGA I is stuffed, 0 otherwise Bit 9 – 1 if FPGA A is 2vp100, 0 if 2vp70 Bit 10 – 1 if FPGA B is 2vp100, 0 if 2vp70 Bit 11 – 1 if FPGA C is 2vp100, 0 if 2vp70 Bit 12 – 1 if FPGA D is 2vp100, 0 if 2vp70 Bit 13 – 1 if FPGA E is 2vp100, 0 if 2vp70 Bit 14 – 1 if FPGA F is 2vp100, 0 if 2vp70 Bit 15 – 1 if FPGA G is 2vp100, 0 if 2vp70 Bit 16 – 1 if FPGA H is 2vp100, 0 if 2vp70

FPGA B				
	Start Address	End Address	Read / Write	Description
				Bit 17 – 1 if FPGA I is 2vp100, 0 if 2vp70

FPGA C				
	Start Address	End Address	Read / Write	Description
DDR 1 (U28)	0x2000_0000	0x20FF_FFFF	R/W	Address maps directly to DDR 1 (32Mx16). If larger DDRs are installed then the address space just extends upward and the start address for DDR2 is moved upward by the same amount.
DDR 2 (U37)	0x2100_0000	0x21FF_FFFF	R/W	Address maps directly to DDR 2 (U37 only avail if 2vp100). If larger DDRs are installed then the address space just extends upward.
DDR Phase Shift Register	0x2C00_0000	0x2C00_0000	R/W	DDR phase shift value (upper WORD is read only and contains the current phase shift value, lower WORD is write only)
External Host Commands Register	0x2C00_0004	0x2C00_0004	R/W	Write/Read register for MCU to issue the following commands: 0x1 – test all functionality 0x2 – test registers 0x4 – test DDR(s) 0x5 – test FPGA interconnect To issue a command the MCU must write one of the above values to this register. The MCU can then poll this register to check if the test is done (register will return all zeros when finished)
Status Register	0x2C00_0008	0x2C00_0008	R	Read-only register - Status of commands and bus control: Bits 16-18 must all be zero for MCU to issue any commands to External Host Command Register. Status results can be read after command register is cleared. The decode for the test results is as follows: Bit 0 – overall pass/fail (pass = 1, fail = 0) Bit 1 – register test pass/fail Bit 3 – ddr test pass/fail Bit 4 – interconnect test pass/fail
Existing FPGA Register	0x2C00_0018	0x2C00_0018	R/W	Contains information on what FPGAs are stuffed on the DN6000K10PCI as well as what type of FPGAs they are. The register has the following format: Bit 0 – 1 if FPGA A is stuffed, 0 otherwise Bit 1 – 1 if FPGA B is stuffed, 0 otherwise Bit 2 – 1 if FPGA C is stuffed, 0 otherwise Bit 2 – 1 if FPGA D is stuffed, 0 otherwise Bit 2 – 1 if FPGA E is stuffed, 0 otherwise Bit 2 – 1 if FPGA F is stuffed, 0 otherwise Bit 2 – 1 if FPGA G is stuffed, 0 otherwise Bit 7 – 1 if FPGA H is stuffed, 0 otherwise Bit 8 – 1 if FPGA I is stuffed, 0 otherwise Bit 9 – 1 if FPGA A is 2vp100, 0 if 2vp70 Bit 10 – 1 if FPGA B is 2vp100, 0 if 2vp70 Bit 11 – 1 if FPGA C is 2vp100, 0 if 2vp70 Bit 12 – 1 if FPGA D is 2vp100, 0 if 2vp70 Bit 13 – 1 if FPGA E is 2vp100, 0 if 2vp70 Bit 14 – 1 if FPGA F is 2vp100, 0 if 2vp70 Bit 15 – 1 if FPGA G is 2vp100, 0 if 2vp70 Bit 16 – 1 if FPGA H is 2vp100, 0 if 2vp70 Bit 17 – 1 if FPGA I is 2vp100, 0 if 2vp70

FPGA D

	Start Address	End Address	Read / Write	Description
DDR 1 (U29)	0x3000_0000	0x30FF_FFFF	R/W	Address maps directly to DDR 1 (32Mx16). If larger DDRs are installed then the address space just extends upward and the start address for DDR2 is moved upward by the same amount.
DDR 2 (U38)	0x3100_0000	0x31FF_FFFF	R/W	Address maps directly to DDR 2 (U34 only avail if 2vp100). If larger DDRs are installed then the address space just extends upward.
DDR Phase Shift Register	0x3C00_0000	0x3C00_0000	R/W	DDR phase shift value (upper WORD is read only and contains the current phase shift value, lower WORD is write only)
External Host Commands Register	0x3C00_0004	0x3C00_0004	R/W	Write/Read register for MCU to issue the following commands: 0x1 – test all functionality 0x2 – test registers 0x4 – test DDR(s) 0x5 – test FPGA interconnect To issue a command the MCU must write one of the above values to this register. The MCU can then poll this register to check if the test is done (register will return all zeros when finished)
Status Register	0x3C00_0008	0x3C00_0008	R	Read-only register - Status of commands and bus control: Bits 16-18 must all be zero for MCU to issue any commands to External Host Command Register. Status results can be read after command register is cleared. The decode for the test results is as follows: Bit 0 – overall pass/fail (pass = 1, fail = 0) Bit 1 – register test pass/fail Bit 3 – ddr test pass/fail Bit 4 – interconnect test pass/fail
Existing FPGA Register	0x3C00_0018	0x3C00_0018	R/W	Contains information on what FPGAs are stuffed on the DN6000k10PCI as well as what type of FPGAs they are. The register has the following format: Bit 0 – 1 if FPGA A is stuffed, 0 otherwise Bit 1 – 1 if FPGA B is stuffed, 0 otherwise Bit 2 – 1 if FPGA C is stuffed, 0 otherwise Bit 2 – 1 if FPGA D is stuffed, 0 otherwise Bit 2 – 1 if FPGA E is stuffed, 0 otherwise Bit 2 – 1 if FPGA F is stuffed, 0 otherwise Bit 2 – 1 if FPGA G is stuffed, 0 otherwise Bit 7 – 1 if FPGA H is stuffed, 0 otherwise Bit 8 – 1 if FPGA I is stuffed, 0 otherwise Bit 9 – 1 if FPGA A is 2vp100, 0 if 2vp70 Bit 10 – 1 if FPGA B is 2vp100, 0 if 2vp70 Bit 11 – 1 if FPGA C is 2vp100, 0 if 2vp70 Bit 12 – 1 if FPGA D is 2vp100, 0 if 2vp70 Bit 13 – 1 if FPGA E is 2vp100, 0 if 2vp70 Bit 14 – 1 if FPGA F is 2vp100, 0 if 2vp70 Bit 15 – 1 if FPGA G is 2vp100, 0 if 2vp70 Bit 16 – 1 if FPGA H is 2vp100, 0 if 2vp70 Bit 17 – 1 if FPGA I is 2vp100, 0 if 2vp70

FPGA E

	Start Address	End Address	Read / Write	Description
DDR 1 (U49)	0x4000_0000	0x40FF_FFFF	R/W	Address maps directly to DDR 1 (32Mx16). If larger DDRs are installed then the address space just extends upward and the start address for DDR2 is moved upward by the same amount.
DDR 2 (U58)	0x4100_0000	0x41FF_FFFF	R/W	Address maps directly to DDR 2 (U58 only avail if 2vp100). If larger DDRs are installed then the address space just extends upward.
SRAM (U71)	0x4800_0000	0x4807_FFFF	R/W	Address maps directly to SRAM. If larger SRAMs are installed then the address space just extends upward.
DDR Phase Shift Register	0x4C00_0000	0x4C00_0000	R/W	DDR phase shift value (upper WORD is read only and contains the current phase shift value, lower WORD is write only)
External Host Commands Register	0x4C00_0004	0x4C00_0004	R/W	Write/Read register for MCU to issue the following commands: 0x1 – test all functionality 0x2 – test registers 0x3 – test SRAM 0x4 – test DDR(s) 0x5 – test FPGA interconnect To issue a command the MCU must write one of the above values to this register. The MCU can then poll this register to check if the test is done (register will return all zeros when finished)
Status Register	0x4C00_0008	0x4C00_0008	R	Read-only register - Status of commands and bus control: Bits 16-18 must all be zero for MCU to issue any commands to External Host Command Register. Status results can be read after command register is cleared. The decode for the test results is as follows: Bit 0 – overall pass/fail (pass = 1, fail = 0) Bit 1 – register test pass/fail Bit 2 – flash test pass/fail Bit 3 – ddr test pass/fail Bit 4 – interconnect test pass/fail
Existing FPGA Register	0x4C00_0018	0x4C00_0018	R/W	Contains information on what FPGAs are stuffed on the DN6000k10PCI as well as what type of FPGAs they are. The register has the following format: Bit 0 – 1 if FPGA A is stuffed, 0 otherwise Bit 1 – 1 if FPGA B is stuffed, 0 otherwise Bit 2 – 1 if FPGA C is stuffed, 0 otherwise Bit 2 – 1 if FPGA D is stuffed, 0 otherwise Bit 2 – 1 if FPGA E is stuffed, 0 otherwise Bit 2 – 1 if FPGA F is stuffed, 0 otherwise Bit 2 – 1 if FPGA G is stuffed, 0 otherwise Bit 7 – 1 if FPGA H is stuffed, 0 otherwise Bit 8 – 1 if FPGA I is stuffed, 0 otherwise Bit 9 – 1 if FPGA A is 2vp100, 0 if 2vp70 Bit 10 – 1 if FPGA B is 2vp100, 0 if 2vp70 Bit 11 – 1 if FPGA C is 2vp100, 0 if 2vp70 Bit 12 – 1 if FPGA D is 2vp100, 0 if 2vp70 Bit 13 – 1 if FPGA E is 2vp100, 0 if 2vp70 Bit 14 – 1 if FPGA F is 2vp100, 0 if 2vp70 Bit 15 – 1 if FPGA G is 2vp100, 0 if 2vp70 Bit 16 – 1 if FPGA H is 2vp100, 0 if 2vp70 Bit 17 – 1 if FPGA I is 2vp100, 0 if 2vp70

FPGA F

	Start Address	End Address	Read / Write	Description
DDR 1 (U47)	0x5000_0000	0x50FF_FFFF	R/W	Address maps directly to DDR 1 (32Mx16). If larger DDRs are installed then the address space just extends upward and the start address for DDR2 is moved upward by the same amount.
DDR 2 (U57)	0x5100_0000	0x51FF_FFFF	R/W	Address maps directly to DDR 2 (U57 only avail if 2vp100). If larger DDRs are installed then the address space just extends upward.
SRAM (U70)	0x5800_0000	0x5807_FFFF	R/W	Address maps directly to SRAM. If larger SRAMs are installed then the address space just extends upward.
DDR Phase Shift Register	0x5C00_0000	0x5C00_0000	R/W	DDR phase shift value (upper WORD is read only and contains the current phase shift value, lower WORD is write only)
External Host Commands Register	0x5C00_0004	0x5C00_0004	R/W	Write/Read register for MCU to issue the following commands: 0x1 – test all functionality 0x2 – test registers 0x3 – test SRAM 0x4 – test DDR(s) 0x5 – test FPGA interconnect To issue a command the MCU must write one of the above values to this register. The MCU can then poll this register to check if the test is done (register will return all zeros when finished)
Status Register	0x5C00_0008	0x5C00_0008	R	Read-only register - Status of commands and bus control: Bits 16-18 must all be zero for MCU to issue any commands to External Host Command Register. Status results can be read after command register is cleared. The decode for the test results is as follows: Bit 0 – overall pass/fail (pass = 1, fail = 0) Bit 1 – register test pass/fail Bit 2 – sram test pass/fail Bit 3 – ddr test pass/fail Bit 4 – interconnect test pass/fail
Existing FPGA Register	0x5C00_0018	0x5C00_0018	R/W	Contains information on what FPGAs are stuffed on the DN6000k10PCI as well as what type of FPGAs they are. The register has the following format: Bit 0 – 1 if FPGA A is stuffed, 0 otherwise Bit 1 – 1 if FPGA B is stuffed, 0 otherwise Bit 2 – 1 if FPGA C is stuffed, 0 otherwise Bit 2 – 1 if FPGA D is stuffed, 0 otherwise Bit 2 – 1 if FPGA E is stuffed, 0 otherwise Bit 2 – 1 if FPGA F is stuffed, 0 otherwise Bit 2 – 1 if FPGA G is stuffed, 0 otherwise Bit 7 – 1 if FPGA H is stuffed, 0 otherwise Bit 8 – 1 if FPGA I is stuffed, 0 otherwise Bit 9 – 1 if FPGA A is 2vp100, 0 if 2vp70 Bit 10 – 1 if FPGA B is 2vp100, 0 if 2vp70 Bit 11 – 1 if FPGA C is 2vp100, 0 if 2vp70 Bit 12 – 1 if FPGA D is 2vp100, 0 if 2vp70 Bit 13 – 1 if FPGA E is 2vp100, 0 if 2vp70 Bit 14 – 1 if FPGA F is 2vp100, 0 if 2vp70 Bit 15 – 1 if FPGA G is 2vp100, 0 if 2vp70 Bit 16 – 1 if FPGA H is 2vp100, 0 if 2vp70

FPGA F				
	Start Address	End Address	Read / Write	Description
				Bit 17 – 1 if FPGA I is 2vp100, 0 if 2vp70

Appendix B - AETEST

1 AETEST Installation Instructions

1.1 DOS and Windows 95/98/ME using DPMI

Precompiled executables, **aetestdj.exe** and **cwsdpmi.exe**, are included in the CD-ROM which is shipped with your DN6000K10S Logic Emulation board. If the user is running DOS on a Windows 95/98/ME machine, the PC must be booted using a DOS boot disk. A DOS boot disk is packaged with the DN6000K10S. The user only needs to follow the steps listed below to run the DPMI version of AETEST.

Follow the procedures listed below for installation:

1. Place the files **aetestdj.exe** and **cwsdpmi.exe** (The DOS Extender) into the same directory on your PC/machine.
2. Boot into DOS mode - if you have not already done so.
3. A DOS Boot disk must be used on the Windows machine.
4. Run **aetestdj.exe**.

1.2 Windows 98/ME using a VxD driver

Instead of running AETEST directly from DOS, the user can run AETEST with a VxD device driver. The driver file **PCICFG.VXD** and the executable **aetest98.exe** are included on the DN6000K10S CD-ROM. The driver's source code and its makefile are also included.

Follow the procedures listed below for installation:

1. Place **PCICFG.VXD** and **aetest98.exe** into the same directory.
2. When Windows first starts with the device plugged in, it should ask for a device driver. Select "**Specify the location of the driver**". Note that the board must be configured with a valid bitfile. Our reference design will work.
3. Select "**Display a list of the drivers in a specific location...**".
4. Select "**Other devices**".
5. Under the "**Manufacturers**" tab, select "**unknown device**".

6. Under “**Models**”, select “**unsupported device**”.
7. Run **aetest98.exe**.

NOTE: To re-compile the driver file **PCICFG.VXD**, the user must download the VtoolsD compiler from <http://www.numega.com/>.

1.3 Windows 2000/XP

The precompiled executable **aetest_wdm.exe** and its source code are included in the DN6000K10S CD-ROM. The driver file **DnDev.sys** and its corresponding inf file are also included in the CD-ROM.

Follow the procedures listed below for installation:

1. If the old version of AETEST’s NT driver is installed on the machine, it must be uninstalled.
2. Start the PC with the DN6000K10S plugged – Windows should recognize the board and ask for a driver. Note that the board must be configured with a valid bitfile. Our reference design will work.
3. When the “**Found New Hardware Wizard**” box pops up, click “**Next**”.
4. Select “**Display a list of the known drivers for this device so that I can choose a specific driver**”.
5. Select “**Other device**”.
6. Select “**Have Disk**”.
7. Go to the directory where “**Dndev.inf**” is located (**Source Code\PCI_Software\wdmdrv\drv**) and select it.
8. Locate the driver file “**DnDev.sys**”, under the directory **Source Code\PCI_Software\wdmdrv\ drv\objchk\i386**.
9. Click on one of the devices and select “**Next**”.
10. Run **aetest_wdm.exe**.

NOTE: To compile **aetest_wdm.exe**, the user must use Visual C++ 6.0. setupapi.lib in version 5.0 does not contain all of the necessary functions.

1.4 Windows NT

Precompiled executables, **aetestnt.exe** and **install.exe**, are included in the CD-ROM which is shipped with your DN6000K10S Logic Emulation board.

The driver files **QLDriver.sys** and **QLDriver_16MB.sys** are also included in the CD-ROM, located under **Source Code\PCI_Software\ntdriver\driver\I386\checked**. The two driver files are identical except **QLDriver_16MB.sys** only allocates a maximum of 16MB per BAR. It is useful for systems with insufficient RAM. To use it, rename it to **QLDriver.sys** and re-install.

Follow the procedures listed below for installation:

1. Place the files **install.exe** and **QLDriver.sys** into the same directory on your PC/machine.
2. Type “**install**”.
3. After the driver is installed, start the driver by selecting **Control Panel→Devices→find “QLDriver”→click “Start”**
4. Run **aetestnt.exe**.

Note: Although this driver will work under Windows 2000, we recommend using the WDM driver instead. If you must use it, see the README.txt file in the ../ntdriver/docs directory on the CD-ROM.

1.5 Linux

This version of AETEST has been tested on Red Hat Linux 7.2 (kernel version 2.4.x).

The driver file **dndev.o** and its source code are included in the DN6000K10S CD-ROM. The scripts **dndev_load** and **dndev_unload**, which are also included in the CD-ROM, are used to load and unload the driver.

Follow the procedures listed below for installation:

1. Login as root to start the driver and run the program.
2. Load the driver; type “**sh dndev_load**”.
3. Unload the driver; type “**sh dndev_unload**”.
4. After the driver is loaded, run the utility **aetest_linux**.
5. The user may need to run chmod on **aetest_linux** to make it executable: type “**chmod u+x aetest_linux**”.

NOTE: All text files including scripts are DOS text format (with an extra carriage return character after every new line), they must be converted.

1.6 Solaris

The utility and driver are tested on Solaris 7.0/Sparc with the 32-bit kernel.

Follow the procedures listed below for installation:

1. Login as root to install and run AETEST
2. Go to the driver directory, make sure the driver file “**dndev**” is in the sparc sub-directory.
3. To install the driver, run “**sh dndev_install.sh**”.
4. To uninstall the driver, run “**sh dndev_uninstall.sh**”.
5. Run **aetest_solaris**
6. The user may need to run chmod on **aetest_solaris** to make it executable: type “**chmod u+x aetest_solaris**”.

The driver is compiled with the gcc compiler.

aetest_solaris is compiled with “gmake”. You can download it from the GNU website. The “make” from the Solaris installation does not work with our makefile format.

NOTE: All text files, including scripts are DOS text format (with an extra carriage return character after every new line) they must be converted.

2 AETEST Basic C++ Functions

The AETEST utility program is built on a core of basic C++ functions. These functions perform a variety of PCI accesses (e.g. configuration reads/writes, memory read/writes) and test functions (e.g. memory tests). This appendix will describe a handful of these functions.

2.1 bar_write_byte

bar_write_byte is a high-level function C++ function which is recommended for development by users of the DN6000K10S.

2.1.1 Description

bar_write_byte allows users of the DN6000K10S to write a byte of data to any location in the Base Address Registers (BARs) of PCI memory. All 4 gigabytes of PCI memory is available for access.

2.1.2 Arguments

The arguments for **bar_write_byte** are shown in [Table 32](#). They are listed in order.

Table 32: **bar_write_byte** Arguments

Argument	Description	Possible Values
unsigned long barnum	BAR number to be accessed	BAR0 = 0, BAR1 = 1, BAR2 = 2, BAR3 = 3, BAR4 = 4, or BAR5 = 5
unsigned long byte_offset	Address - Number of bytes to offset data	0x0 – bytes in BAR's mem. space
byte ¹ data	A byte of data for the write operation (8 bits)	0x00 – 0xff

¹typedef unsigned char byte;

2.1.3 Return Values

A successful function call will return zero.

2.1.4 Notes

The source code for **bar_write_byte** is portable to each of the operating systems intended for AETEST usage.

2.2 bar_write_word

bar_write_word is a high-level function C++ function which is recommended for development by users of the DN6000K10S.

2.2.1 Description

bar_write_word allows users of the DN6000K10S to write a word of data to any location in the Base Address Registers (BARs) of PCI memory. All 4 gigabytes of PCI memory is available for access.

2.2.2 Arguments

The arguments for **bar_write_word** are shown in [Table 33](#). They are listed in order.

Table 33: **bar_write_word** Arguments

Argument	Description	Possible Values
unsigned long barnum	BAR number to be accessed	BAR0 = 0, BAR1 = 1, BAR2 = 2, BAR3 = 3, BAR4 = 4, or BAR5 = 5
unsigned long byte_offset	Address - Number of bytes to offset data	0x0 – bytes in BAR's mem. space
word ¹ data	A word of data for the write operation (16 bits)	0x0000 – 0xffff

¹typedef unsigned char word;

2.2.3 Return Values

A successful function call will return zero.

2.2.4 Notes

The source code for **bar_write_word** is portable to each of the operating systems intended for AETEST usage.

2.3 bar_write_dword

bar_write_dword is a high-level function C++ function which is recommended for development by users of the DN6000K10S.

2.3.1 Description

bar_write_dword allows users of the DN6000K10S to write a dword of data to any location in the Base Address Registers (BARs) of PCI memory. All 4 gigabytes of PCI memory is available for access.

2.3.2 Arguments

The arguments for **bar_write_dword** are shown in [Table 34](#). They are listed in order.

Table 34: **bar_write_dword** Arguments

Argument	Description	Possible Values
unsigned long barnum	BAR number to be accessed	BAR0 = 0, BAR1 = 1, BAR2 = 2, BAR3 = 3, BAR4 = 4, or BAR5 = 5
unsigned long byte_offset	Address - Number of bytes to offset data	0x0 – bytes in BAR's mem. space
dword ¹ data	A dword of data for the write operation (32 bits)	0x00000000 – 0xffffffff

¹typedef unsigned char dword;

2.3.3 Return Values

A successful function call will return zero.

2.3.4 Notes

The source code for **bar_write_dword** is portable to each of the operating systems intended for AETEST usage.

2.4 bar_read_byte

bar_read_byte is a high-level function C++ function which is recommended for development by users of the DN6000K10S.

2.4.1 Description

bar_read_byte allows users of the DN6000K10S to read a byte of data from any location in the Base Address Registers (BARs) of PCI memory. All 4 gigabytes of PCI memory is available for access.

2.4.2 Arguments

The arguments for **bar_read_byte** are shown in [Table 35](#). They are listed in order.

Table 35: **bar_read_byte** Arguments

Argument	Description	Possible Values
unsigned long barnum	BAR number to be accessed	BAR0 = 0, BAR1 = 1, BAR2 = 2, BAR3 = 3, BAR4 = 4, or BAR5 = 5
unsigned long byte_offset	Address - Number of bytes to offset data	0x0 – bytes in BAR's mem. space
byte* ¹ data	Pointer to a byte of data for the read operation (8 bits)	0x00 – 0xff

¹typedef unsigned char byte;

2.4.3 Return Values

A successful function call will return zero.

The byte of data read during the access is placed in the variable location pointed to by *data*.

2.4.4 Notes

The source code for **bar_read_byte** is portable to each of the operating systems intended for AETEST usage.

2.5 bar_read_word

bar_read_word is a high-level function C++ function which is recommended for development by users of the DN6000K10S.

2.5.1 Description

bar_read_word allows users of the DN6000K10S to read a word of data from any location in the Base Address Registers (BARs) of PCI memory. All 4 gigabytes of PCI memory is available for access.

2.5.2 Arguments

The arguments for **bar_read_word** are shown in [Table 36](#). They are listed in order.

Table 36: **bar_read_word** Arguments

Argument	Description	Possible Values
unsigned long barnum	BAR number to be accessed	BAR0 = 0, BAR1 = 1, BAR2 = 2, BAR3 = 3, BAR4 = 4, or BAR5 = 5
unsigned long byte_offset	Address - Number of bytes to offset data	0x0 – bytes in BAR's mem. space
word* ¹ data	Pointer to a word of data for the read operation (16 bits)	0x0000 – 0xffff

¹typedef unsigned char word;

2.5.3 Return Values

A successful function call will return zero.

The word of data read during the access is placed in the variable location pointed to by *data*.

2.5.4 Notes

The source code for **bar_read_word** is portable to each of the operating systems intended for AETEST usage.

2.6 bar_read_dword

bar_read_dword is a high-level function C++ function which is recommended for development by users of the DN6000K10S.

2.6.1 Description

bar_read_dword allows users of the DN6000K10S to read a dword of data from any location in the Base Address Registers (BARs) of PCI memory. All 4 gigabytes of PCI memory is available for access.

2.6.2 Arguments

The arguments for **bar_read_dword** are shown in [Table 37](#). They are listed in order.

Table 37: **bar_read_dword** Arguments

Argument	Description	Possible Values
unsigned long barnum	BAR number to be accessed	BAR0 = 0, BAR1 = 1, BAR2 = 2, BAR3 = 3, BAR4 = 4, or BAR5 = 5
unsigned long byte_offset	Address - Number of bytes to offset data	0x0 – bytes in BAR's mem. space
dword*1 data	Pointer to a dword of data for the read operation (32 bits)	0x00000000 – 0xffffffff

¹typedef unsigned char dword;

2.6.3 Return Values

A successful function call will return zero.

The dword of data read during the access is placed in the variable location pointed to by *data*.

2.6.4 Notes

The source code for **bar_read_dword** is portable to each of the operating systems intended for AETEST usage.

2.7 dma_buffer_allocate

dma_buffer_allocate is a high-level function C++ function which is recommended for development by users of the DN6000K10S.

2.7.1 Description

dma_buffer_allocate allows users of the DN6000K10S to allocate a DMA buffer.

2.7.2 Arguments

The arguments for **dma_buffer_allocate** are shown in [Table 38](#). They are listed in order.

Table 38: **dma_buffer_allocate** Arguments

Argument	Description
dma_buffer_handle* ¹ hndl	Pointer to a handle (int) for the allocated DMA buffer
int nbytes	Number of bytes of memory to allocate
int* phy_addr	Pointer to an int specifying the physical address of the DMA buffer

¹typedef int dma_buffer_handle;

2.7.3 Return Values

A successful function call will return zero. An error will return a non-zero value. If -1 is returned, the allocation failed. If -2 is returned, the DPMI implementation of AETEST is not being used (See [Notes](#)).

An integer indicating the handle for the DMA buffer is placed in the variable location pointed to by *hndl*.

An integer indicating the physical address of the DMA buffer is placed in the variable location pointed to by *phy_addr*.

2.7.4 Notes

The **dma_buffer_allocate** code is written for use in the DPMI (DOS) implementation of AETEST.

2.8 dma_buffer_free

dma_buffer_free is a high-level function C++ function which is recommended for development by users of the DN6000K10S.

2.8.1 Description

dma_buffer_free allows users of the DN6000K10S to free memory associated with a previously allocated DMA buffer.

2.8.2 Arguments

The argument(s) for **dma_buffer_free** are shown in [Table 39](#). They are listed in order.

Table 39: **dma_buffer_free** Arguments

Argument	Description
dma_buffer_handle ¹ hndl	Handle for a DMA buffer

¹typedef int dma_buffer_handle;

2.8.3 Return Values

A successful function call will return zero. If -2 is returned, the DPMI implementation of AETEST is not being used (See [Notes](#)).

2.8.4 Notes

The **dma_buffer_free** code is written for use in the DPMI (DOS) implementation of AETEST.

2.9 dma_write_dword

dma_write_dword is a high-level function C++ function which is recommended for development by users of the DN6000K10S.

2.9.1 Description

dma_write_dword allows users of the DN6000K10S to write a dword of data to any byte-aligned location in a DMA buffer.

2.9.2 Arguments

The arguments for **dma_write_dword** are shown in [Table 40](#). They are listed in order.

Table 40: **dma_write_dword** Arguments

Argument	Description
dma_buffer_handle ¹ hndl	Handle for a DMA buffer
int offset	Offset in bytes of the write location in the DMA buffer
dword ² data	A dword (32 bit) of data for the write operation

¹typedef int dma_buffer_handle;

²typedef unsigned char dword;

2.9.3 Return Values

A successful function call will return zero. If -2 is returned, the DPMI implementation of AETEST is not being used (See [Notes](#)).

2.9.4 Notes

The **dma_write_dword** code is written for use in the DPMI (DOS) implementation of AETEST.

2.10 dma_read_dword

dma_read_dword is a high-level function C++ function which is recommended for development by users of the DN6000K10S.

2.10.1 Description

dma_read_dword allows users of the DN6000K10S to read a dword of data from any byte-aligned location in a DMA buffer.

2.10.2 Arguments

The arguments for **dma_read_dword** are shown in [Table 41](#). They are listed in order.

Table 41: **dma_read_dword** Arguments

Argument	Description
dma_buffer_handle ¹ hndl	Handle for a DMA buffer
int offset	Offset in bytes of the write location in the DMA buffer
dword* ² data	Pointer to a dword (32 bit) of data for the read operation

¹typedef int dma_buffer_handle;

²typedef unsigned char dword;

2.10.3 Return Values

A successful function call will return zero. If -2 is returned, the DPMI implementation of AETEST is not being used (See [Notes](#)).

2.10.4 Notes

The **dma_read_dword** code is written for use in the DPMI (DOS) implementation of AETEST.

2.11 pci_rdwr

pci_rdwr is a function used in older revisions of AETEST. Users of the DN6000K10S are advised to use current functions such as `bar_write_dword` and `bar_read_dword` for development.

2.11.1 Description

pci_rdwr is the primary function for reading and writing to the Base Address Registers (BARs).

2.11.2 Arguments

The arguments for **pci_rdwr** are shown in Table 42. They are listed in order.

Table 42: **pci_rdwr** Arguments

Argument	Description	Possible Values
long barnum	BAR number to be accessed	BAR0 = 0, BAR1 = 1, BAR2 = 2, BAR3 = 3, BAR4 = 4, or BAR5 = 5
long byte_offset	Address - Number of bytes to offset data	0x0 – bytes in BAR's mem. space
long upper_data	The upper 32-bits of data for a 64-bit access	0x00000000 – 0xffffffff
long lower_data	Data (the lower 32-bits of a 64-bit access)	0x00000000 – 0xffffffff
int command	PCI command	MEM_READ (0x6) or MEM_WRITE (0x7)
int be	Byte Enables	0x00 - 0xff DWORD_BYTE_EN (0x0f)
int dwordcount	Number of DWORDs	1 or 2
int verify	Verify (TRUE) or do not verify access (FALSE)	0x0 – 0x1

2.11.3 ReturnValues

When **pci_rdwr** is called with 'MEM_READ' as its command argument, the returned DWORD is placed into the variable `access_memory_dword_read`. The declaration for `access_memory_dword_read` is:

```
Extern unsigned long access_memory_dword_read;
```

2.11.4 Notes

In a typical transaction, the **byte_offset** value will be a multiple of 4 resulting in a DWORD aligned read or write. The PCI command will either be a Memory Read or a Memory Write where MEM_READ and MEM_WRITE are #define definitions used in AETEST. BAR_x, where x = 0-5, are also #define definitions in AETEST. The byte enable **be** is often set to DWORD_BYTE_EN for 32-bit transactions. **dwordcount** is either 1 or 2 indicating a 32-bit or a 64-bit transaction respectively. Finally, the parameter **verify** is set to TRUE when the access is to be verified. If verification is not desired, **verify** is set to FALSE.

2.12 DeviceIoControl

DeviceIoControl is a low-level function. Users of the DN6000K10S are advised to use higher level functions such as `bar_write_dword` and `bar_read_dword` for development.

2.12.1 Description

DeviceIoControl is used to send commands and receive messages from a specified device on the PCI bus in a Windows environment. The QL library is based upon this function.

A successful **DeviceIoControl** operation will return zero. A non-zero value is returned if a failure occurs.

2.12.2 Arguments

The arguments for the **DeviceIoControl** method is listed in [Table 43](#). They are listed in order.

Table 43: **DeviceIoControl** Arguments

Argument	Description
HANDLE hDevice	Handle to the device for operation
DWORD dwIoControlCode	Control code for the operation
LPVOID IpInBuffer	Pointer to a buffer containing data necessary for operation
DWORD nInBufferSize	Specifies the size, in bytes, of the buffer pointed to by IpInBuffer
LPVOID IpOutBuffer	Pointer to a buffer that receives the operation's output data
DWORD nOutBufferSize	Specifies the size, in bytes, of the buffer pointed to by IpOutBuffer
LPDWORD IpBytesReturned	Pointer to a variable that receives the size, in bytes, of the data stored into the buffer pointed to by IpOutBuffer
LPOVERLAPPED IpOverlapped	Pointer to an OVERLAPPED structure

2.12.3 Return Values

A successful **DeviceIoControl** operation will return zero. A non-zero value is returned if a failure occurs.

2.12.4 Notes

hDevice

The CreateFile function should be used to retrieve a handle.

dwIoControlCode

See include file qlcntlcodes.h, which is included with the AETEST source code, for example control codes.

IpInBuffer

This parameter can be set to NULL if no input data is required for the operation.

nInBufferSize

N/A

IpOutBuffer

This parameter can be set to NULL if operation does not produce any output data.

NOutBufferSize

N/A

IpBytesReturned

If the output buffer is too small, the call function fails and the returned byte count is zero. If the output buffer is full prior to operation completion, the call will fail. However, **DeviceIoControl** will return all of the data in the output buffer and returned byte count will correspond to the amount of data returned.

IpOverlapped

If hDevice was opened with the FILE_FLAG_OVERLAPPED flag, IpOverlapped must point to a valid OVERLAPPED structure. Under these conditions, the operation is asynchronous (i.e. an overlapped operation). If IpOverlapped is NULL under these conditions, the function will fail.

If the FILE_FLAG_OVERLAPPED was not used to open hDevice, IpOverlapped is ignored. The operation must complete before DeviceIoControl will return.

2.12.5 Derived Functions

The following functions are based on **DeviceIoControl**:

QL_ConfigRead, QL_ConfigWrite, QL_ControlRead, QL_ControlWrite,
QL_BAR_Read, QL_BAR_Write, QL_MapBufferAddr, QL_UnMapBufferAddr,
QL_GetBufferSize, QL_DMA_Read, QL_DMA_Write, QL_Map_BAR,
QL_UnMap_BAR and QL_ResetDevice.